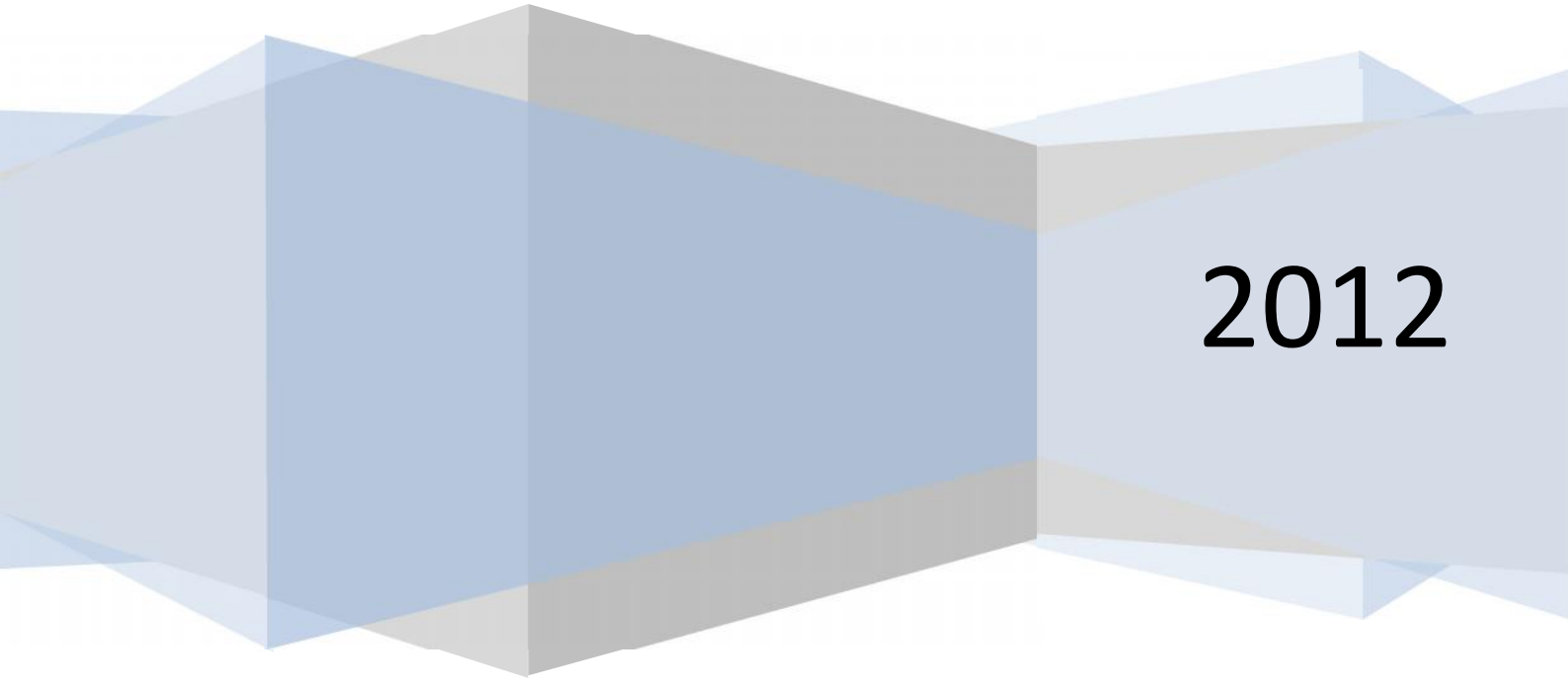
Ecole Polytechnique

Diagramme d'Etat

Extrait de UML 2 Par la Pratique 5 Edition

M. ECHCHADLI



2012

Modélisation dynamique :

■ Acteur ■ Cas d'utilisation, scénario ■ Diagramme de séquence système ■ Diagramme de contexte dynamique ■ Message ■ Diagramme d'états ■ État, transition ■ Événement ■ Condition ■ Action - Activité.

Ce chapitre va nous permettre d'illustrer pas à pas, à partir d'une nouvelle étude de cas, les principaux concepts et diagrammes UML pour le point de vue dynamique.

En commençant par identifier les acteurs et les cas d'utilisation, nous dessinerons tout d'abord un diagramme de séquence « système ». Puis, nous réaliserons un diagramme de communication particulier (que nous appellerons diagramme de contexte dynamique) afin de répertorier tous les messages que les acteurs peuvent envoyer au système et recevoir.

Après ce travail préliminaire, nous nous embarquerons dans une description en profondeur de la dynamique souhaitée du système. Nous insisterons tout particulièrement sur le dia- gramme d'états qui est à notre avis trop souvent sous- employé, alors que c'est un diagramme extrêmement utile pour décrire avec précision des comportements complexes, et ce dès l'analyse.

En complément des concepts de base, nous expliquerons la bonne utilisation des concepts avancés tels que :

- Événement interne : *when*
- État composite et sous-états exclusifs
- Transition propre et transition interne
- Pseudo-état : *History*
- Envoi de message : *send*

PRINCIPES ET DÉFINITIONS DE BASE

DIAGRAMME D'ÉTATS

UML a repris le concept bien connu de *machine à états finis*, qui consiste à s'intéresser au cycle de vie d'une instance générique d'une classe particulière au fil de ses interactions avec le reste du monde, dans tous les cas possibles. Cette vue locale d'un objet, qui décrit comment il réagit à des événements en fonction de son état courant et comment il passe dans un nouvel état, est représentée graphiquement sous la forme d'un *diagramme d'états*.

Un diagramme d'états pour quelles classes ?

Toutes les classes du modèle statique ne requièrent pas nécessairement une machine à états, représentée par un diagramme d'états. Il s'agit donc de trouver celles qui ont un comportement dynamique complexe nécessitant une description poussée. Cela

correspond à l'un des deux cas suivants :

- les objets de la classe peuvent-ils réagir différemment à l'occurrence du même événement ? Chaque type de réaction caractérise un état particulier ;
- la classe doit-elle organiser certaines opérations dans un ordre précis ? Dans ce cas, des états séquentiels permettent de préciser la chronologie forcée des événements d'activation.

Comment le construire ?

1. Représentez tout d'abord la séquence d'états qui décrit le comportement nominal d'un objet, avec les transitions qui y sont associées.
2. Ajoutez progressivement les transitions qui correspondent aux comportements alternatifs ou d'erreur.
3. Complétez les effets sur les transitions et les activités dans les états.
4. Structurez le diagramme en sous-états s'il devient trop complexe.

Comment le représenter ?

Un diagramme d'états est un graphe orienté d'états et de transitions.

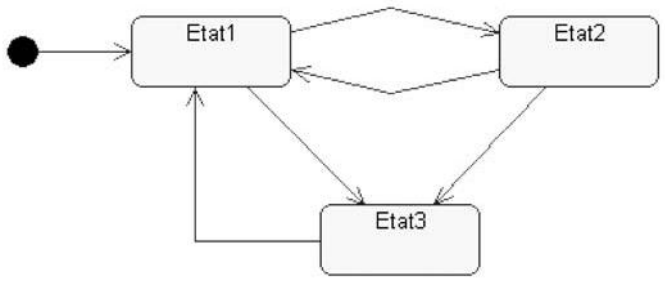


Figure 5-1.
Notation de base du diagramme d'états

ÉTAT

Un *état* représente une situation durant la vie d'un objet pendant laquelle :

- il satisfait une certaine condition ;
- il exécute une certaine activité ;
- ou bien il attend un certain événement.

Un objet passe par une succession d'états durant son existence. Un état a une durée finie, variable selon la vie de l'objet, en particulier en fonction des événements qui lui arrivent.

Comment les identifier ?

Comment trouver les états d'une classe ? Pour faire un parallèle, on peut dire qu'il est aussi difficile de trouver les bons états dans le modèle dynamique que les bonnes classes dans le modèle statique ! Il n'y a donc pas de recette miracle, cependant trois démarches complémentaires peuvent être mises en œuvre :

- la recherche intuitive repose sur l'expertise métier. Certains états fondamentaux font partie du vocabulaire des experts du domaine et sont identifiables a priori ;

- l'étude des attributs et des associations de la classe peut donner des indications précieuses : cherchez des valeurs seuils d'attributs qui modifient la dynamique, ou des comportements qui sont induits par l'existence ou l'absence de certains liens ;
- une démarche systématique peut également être utilisée : classe par classe, cherchez le diagramme d'interaction le plus représentatif du comportement des instances de cette classe, associez un état à chaque intervalle entre événements émis ou reçus par une instance et placez les transitions. Reproduisez ensuite cette démarche avec tous les scénarios faisant intervenir des instances de la classe, afin d'ajouter de nouvelles transitions ou de nouveaux états. La difficulté principale consiste à trouver ensuite les boucles dans le diagramme, afin de ne pas multiplier les états.

État initial et état final

En plus de la succession d'états « normaux » correspondant au cycle de vie d'un objet, le diagramme d'états comprend également deux pseudo-états :

- L'état initial du diagramme d'états correspond à la création de l'instance.
- L'état final du diagramme d'états correspond à la destruction de l'instance.

TRANSITION

Une transition décrit la réaction d'un objet lorsqu'un événement se produit (généralement l'objet change d'état). En règle générale, une transition possède un événement déclencheur, une condition de garde, un effet et un état cible.

ÉVÉNEMENT

En UML, un événement spécifie qu'il s'est passé quelque chose de significatif, localisé dans le temps et dans l'espace. Dans le contexte des machines à états finis, il représente l'occurrence d'un stimulus qui peut déclencher une transition entre états.

UML propose de distinguer quatre sortes d'événements :

- la *réception d'un message* envoyé par un autre objet, ou par un acteur. L'envoi d'un message est en général asynchrone ;
- *l'appel d'une opération* (*call event*) sur l'objet récepteur. L'événement d'appel est en général synchrone ;
- le *passage du temps (time event)*, qui se modélise en utilisant le mot-clé `after` suivi d'une expression représentant une durée, décomptée à partir de l'entrée dans l'état courant ;
- un *changement* dans la satisfaction d'une condition (*change event*). On utilise alors le mot-clé `when`, suivi d'une expression booléenne. L'événement de changement se produit lorsque la condition passe à vrai.

Un événement peut porter des paramètres qui matérialisent le flot d'informations ou de données entre objets.

MESSAGE

Un message est une transmission d'information unidirectionnelle entre deux objets, l'objet émetteur et l'objet récepteur.

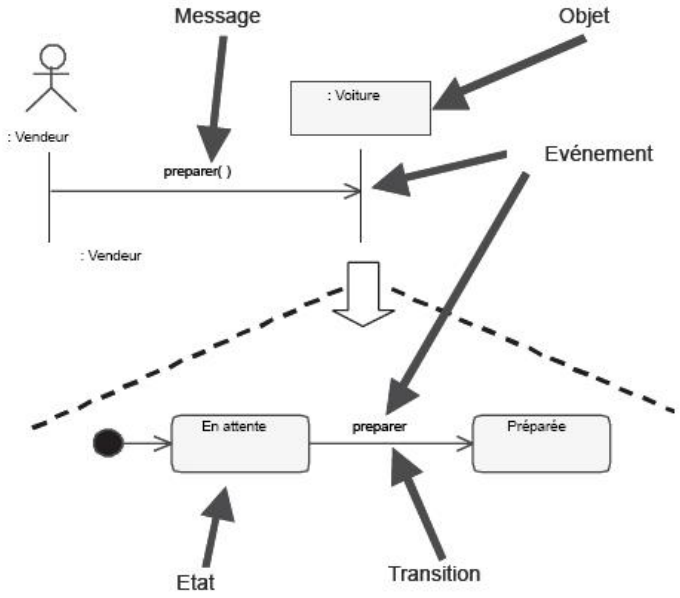
Le mode de communication peut être synchrone ou asynchrone. La réception d'un message est un événement qui est doit être traité par le récepteur.

CONDITION

Une condition (ou condition de garde) est une expression booléenne qui doit être vraie lorsque l'événement arrive pour que la transition soit déclenchée. Elle peut concerner les attributs de l'objet ainsi que les paramètres de l'événement déclencheur. Plusieurs transitions avec le même événement doivent avoir des conditions de garde différentes.

Comment représenter les concepts dynamiques de base ?

Figure 5-2.
Concepts dynamiques de base



EFFET : ACTION OU ACTIVITÉ

Une transition peut spécifier un comportement optionnel réalisé par l'objet lorsque la transition est déclenchée. Ce comportement est appelé « *effet* » en UML 2 : cela peut être une simple action ou une séquence d'actions (une activité). Une action élémentaire[2] peut représenter la mise à jour d'un attribut, un appel d'opération, la création ou la destruction d'un autre objet, ainsi que l'envoi d'un signal à un autre objet. Les *activités* associées aux transitions sont considérées comme atomiques, c'est-à-dire qu'elles ne peuvent être interrompues. Ce sont typiquement des séquences d'actions.

Les *activités durables (do-activity)*, au contraire, ont une certaine durée, sont interruptibles et sont donc associées aux états

COMMENT REPRÉSENTER LES DIAGRAMMES D'ÉTATS ?

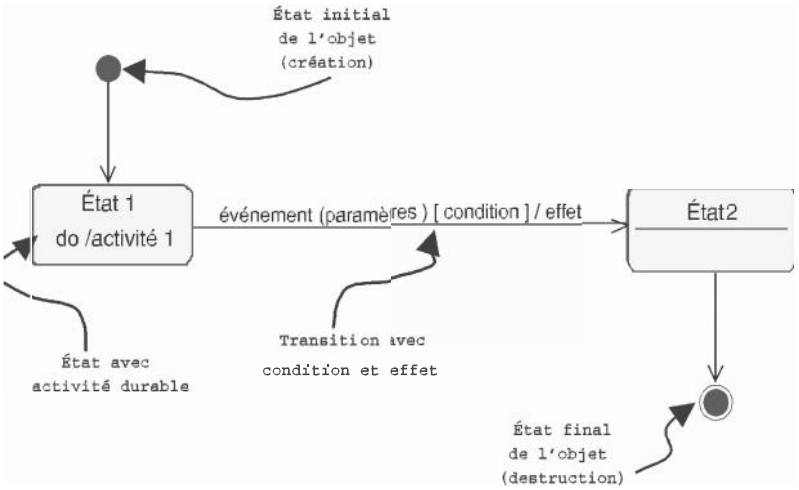


Figure 5-3.
Notation plus complète du diagramme d'états

ÉTUDE D'UN PUBLIPHONE À PIÈCES

Cette étude de cas concerne un système simplifié de Publiphone à pièces.

1. Le prix minimal d'une communication interurbaine est de 0,2 euros.
2. Après l'introduction de la monnaie, l'utilisateur a 2 minutes pour composer son numéro (ce délai est décompté par le standard).
3. La ligne peut être libre ou occupée.
4. Le correspondant peut raccrocher le premier.
5. Le Publiphone consomme de l'argent dès que l'appelé décroche et à chaque unité de temps (UT) générée par le standard.
6. On peut ajouter des pièces à tout moment.
7. Lors du raccrochage, le solde de monnaie est rendu.

À partir de ces six phrases, nous allons progressivement :

1. Identifier les acteurs et les cas d'utilisation.
2. Construire un diagramme de séquence système.
3. Construire le diagramme de contexte dynamique.

Élaborer le diagramme d'états du Publiphone

Étape 1 – Identification des acteurs et des cas d'utilisation

En premier lieu, nous allons identifier les acteurs et les cas d'utilisation du Publiphone à pièces.

Diagramme de cas d'utilisation

Dessinez le diagramme de cas d'utilisation du Publiphone à pièces.

Solution

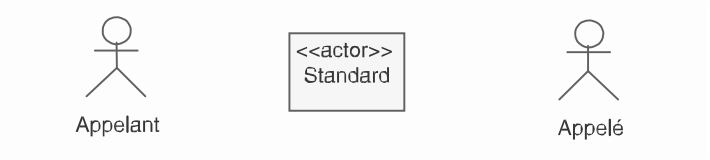
Quelles sont les entités externes qui interagissent directement avec le Publiphone ?

Si nous procédons à une analyse linguistique de l'énoncé, nous obtenons les cinq candidats suivants : utilisateur, standard, correspondant, Publiphone, appelé.

Éliminons tout de suite Publiphone puisqu'il s'agit du système lui-même. En revanche, le standard est bien un acteur (non-humain) connecté directement au système.

La seule difficulté concerne les acteurs humains : utilisateur, correspondant et appelé. Comme les deux derniers termes semblent être synonymes, nous pouvons garder le mot *appelé* et renommer, par souci de symétrie, utilisateur en *appelant*.

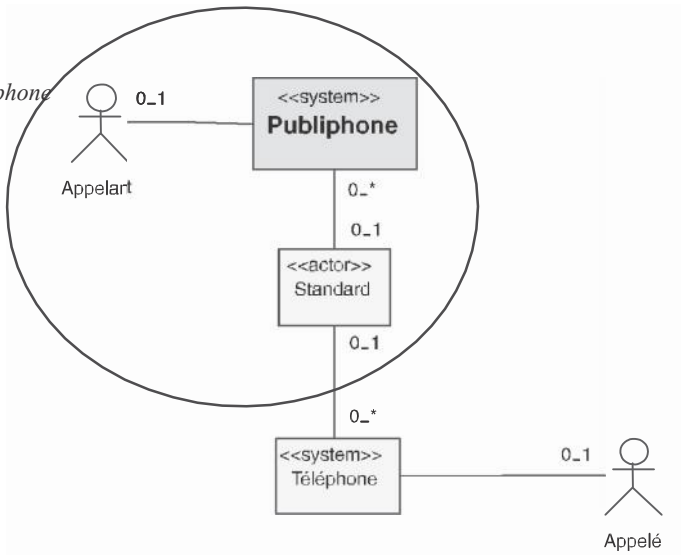
Figure 5-4.
Liste préliminaire des acteurs



Comment les acteurs utilisent-ils le Publiphone ? La seule utilisation vraiment intéressante dans notre contexte est celle de l'appelant qui téléphone à l'appelé. Le standard sert en fait d'intermédiaire entre les deux. Si nous affinons encore notre analyse, nous nous apercevons rapidement que l'appelé n'interagit pas directement avec le Publiphone : il est complètement masqué par le standard.

Une illustration graphique de ce problème de détermination de frontière est donnée sur le diagramme de contexte statique suivant.

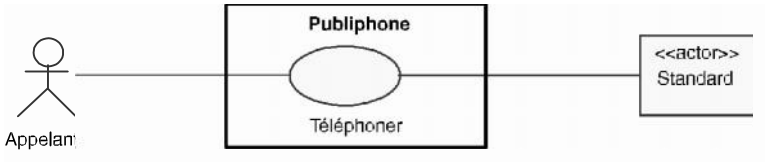
Figure 5-5.
Diagramme de contexte statique élargi du Publiphone



On voit bien sur le diagramme précédent que l'appelant communique avec l'appelé par le biais de trois systèmes connectés : le Publiphone, le standard et le téléphone de l'appelé. On notera la symétrie du schéma en regard du standard, qui joue le rôle d'acteur par rapport aux deux autres systèmes de même nature.

L'appelé est donc un acteur indirect par rapport au Publiphone. Nous ne le retiendrons pas pour notre diagramme de cas d'utilisation qui est finalement très simple.

Figure 5-6.
Diagramme de cas d'utilisation du Publiphone



Étape 2 – Réalisation du diagramme de séquence système

Avant de plonger dans les arcanes du diagramme d'états du Publiphone, nous allons le préparer en réalisant tout d'abord un diagramme de séquence système. Nous avons vu aux chapitres 1 et 2 l'intérêt de ce type de diagramme et les différents détails qui le concernent.

Diagramme de séquence système

Réalisez un diagramme de séquence système qui décrive le scénario nominal du cas d'utilisation TÉLÉPHONER.

Solution

En se basant sur notre connaissance du domaine, nous allons décrire le scénario nominal de succès de communication entre un appelant et un appelé.

Comme cela est expliqué aux chapitres 1 et 2, nous utilisons la convention graphique suivante :

- l'acteur principal *Appelant* à gauche ;
- une ligne de vie représentant le *Publiphone* au milieu ;
- l'acteur secondaire *Standard* à droite.

Notez l'utilisation de deux fragments UML 2 :

- `loop` pour indiquer l'itération sur la phase de conversation avec le transport de la voix et le passage des unités de temps (UT) ;
- `opt` pour indiquer la possibilité d'ajouter des pièces pendant la conversation.

Nous n'avons pas représenté pour l'instant les réponses du Publiphone à l'appelant (en termes de tonalité par exemple) afin de ne pas alourdir ce premier schéma.

Figure 5-7.
Diagramme de séquence système du scénario nominal de Téléphoner

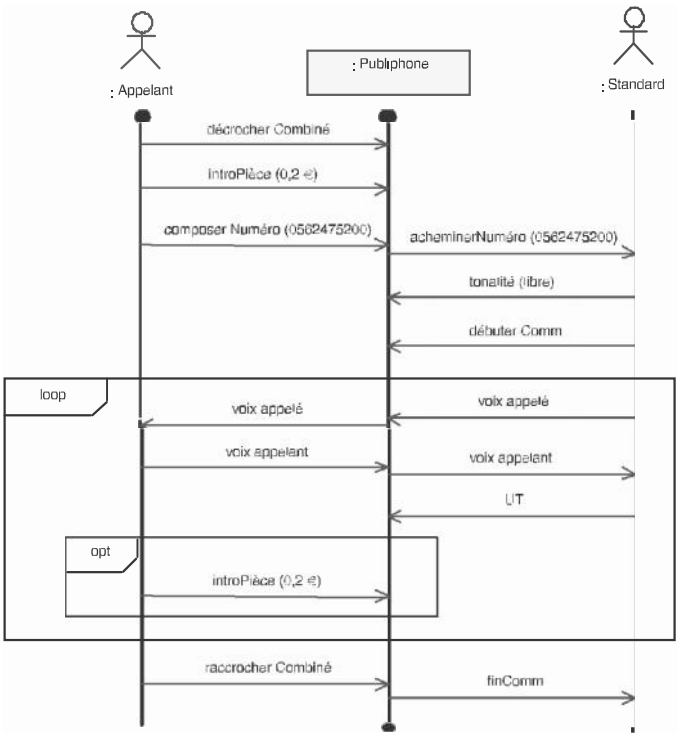


Diagramme de séquence système enrichi

Enrichissez le diagramme de séquence système précédent avec des activités internes intéressantes et quelques réponses du Publiphone à l'appelant.

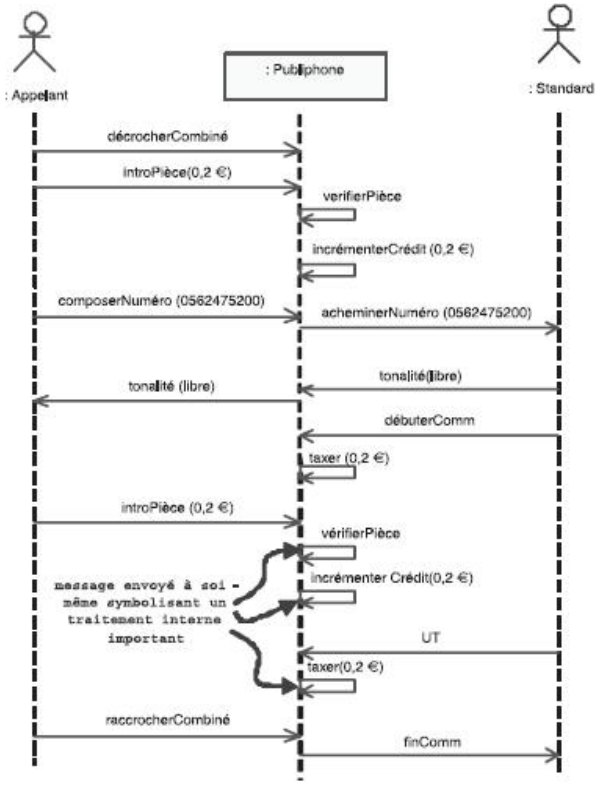
En revanche, ne représentez plus la conversation afin de vous concentrer sur les « opérations système ».

solution

Nous avons ajouté au diagramme précédent les activités internes importantes du Publiphone, telles que la vérification des pièces et la gestion du solde de l'appelant :

- incrémentation lors de l'introduction de pièces ;
- décrémentation lors du début de communication et à chaque UT.

Figure 5-8.
Diagramme de séquence système enrichi du scénario nominal de Téléphoner



Étape 3 – Représentation du contexte dynamique

Pour parfaire la préparation du diagramme d'états, nous allons maintenant répertorier l'ensemble des messages qui sont émis et surtout ceux qui sont reçus par le Publiphone. En effet, les messages reçus vont devenir des événements qui déclenchent des transitions entre états et les messages émis vont donner lieu à des actions sur les transitions.

Le diagramme de séquence système réalisé à l'étape 2 liste un certain nombre de messages. Nous visons maintenant en la matière à l'exhaustivité et à la « généricité ». Dans cette optique, nous préconisons de représenter graphiquement l'ensemble des messages échangés par le système avec ses acteurs sous la forme d'un diagramme de communication appelé *diagramme de contexte dynamique*.

À retenir

REPRÉSENTATION GRAPHIQUE DU CONTEXTE DYNAMIQUE

Utilisez un diagramme de *communication* de la façon suivante :
– le système étudié est représenté par un objet[a] au centre du diagramme ;
– cet objet central est entouré par une instance de chaque acteur ;
– un lien relie le système à chacun des acteurs ;
– sur chaque lien sont répertoriés tous les messages en entrée et en sortie du système, sans numérotation.

Diagramme de contexte dynamique

Réalisez le diagramme de contexte dynamique du Publiphone de la façon indiquée précédemment.

Solution

En partant des deux diagrammes de séquence système, nous avons répertorié les messages échangés entre le système et ses acteurs. Puis nous les avons généralisés en ajoutant des paramètres quand c'était nécessaire :

- introPièce(0,2 €) devient un message paramétré : « introPiece(p) » ;
- composerNumero(0562475200) devient « composerNumero(num) » ;
- tonalité (libre) devient « tonalité(type) » pour prendre en compte l'occupa- tion de la ligne, etc.

Ce premier travail donne le schéma préliminaire suivant :

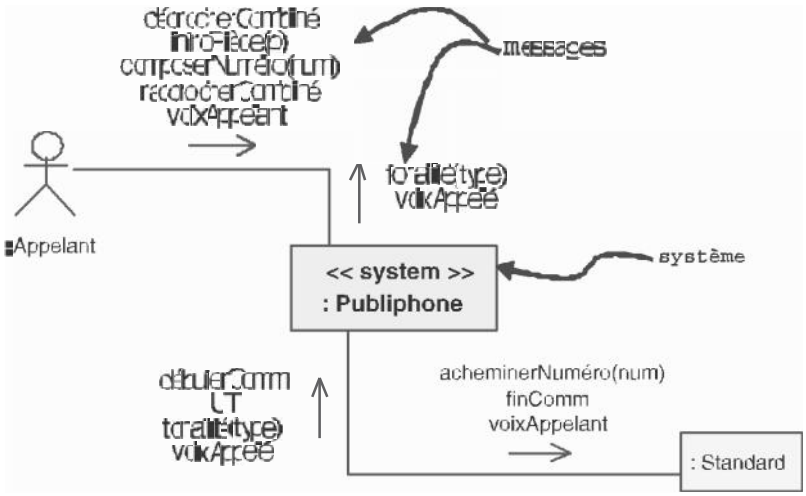


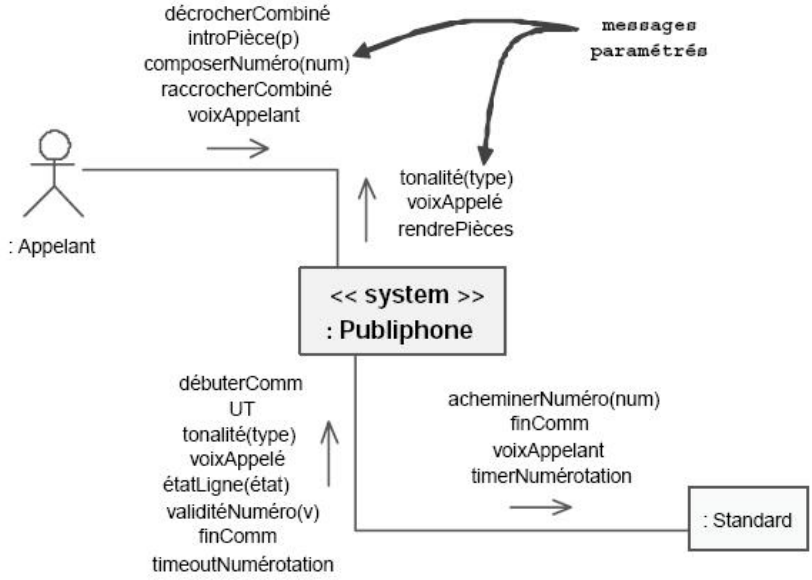
Figure 5-9. *Version préliminaire du diagramme de contexte dynamique*

Toutefois, prenons garde de ne pas oublier que nous sommes partis du diagramme de séquence système qui représente un scénario nominal du cas d'utilisation *Téléphoner*. D'autres messages peuvent être envisagés entre le Publiphone et ses acteurs :

- s'il reste des pièces inutilisées quand l'appelant raccroche, le Publiphone les lui rend ;
- après l'introduction de la somme minimale de 0,2 €, le Publiphone transmet un message au standard pour le décompte du délai de 2 minutes ;
- si le numéro composé n'est pas valide, le standard le détecte ;
- si l'appelé raccroche le premier, la fin de communication est signalée par le standard ;
- plus généralement, le standard transmet l'état de la ligne au Publiphone (libre, occupée, en dérangement, etc.), et pas seulement le type de tonalité.

Le diagramme de contexte dynamique est donc complété comme cela apparaît sur la figure ci-après.

Figure 5-10. *Version complète du diagramme de contexte dynamique*



Étape 4 – Description exhaustive par un diagramme d'états

Après tout ce travail préliminaire, nous pouvons maintenant entreprendre une description exhaustive de la dynamique du Publiphone.

Le comportement du Publiphone n'est pas trivial, comme en témoigne par exemple le nombre élevé de messages identifiés sur le diagramme de contexte dynamique. Nous préconisons dans ce cas une approche itérative et incrémentale. C'est la démarche que nous allons mettre en œuvre au travers des questions qui suivent.

Diagramme d'états préliminaire

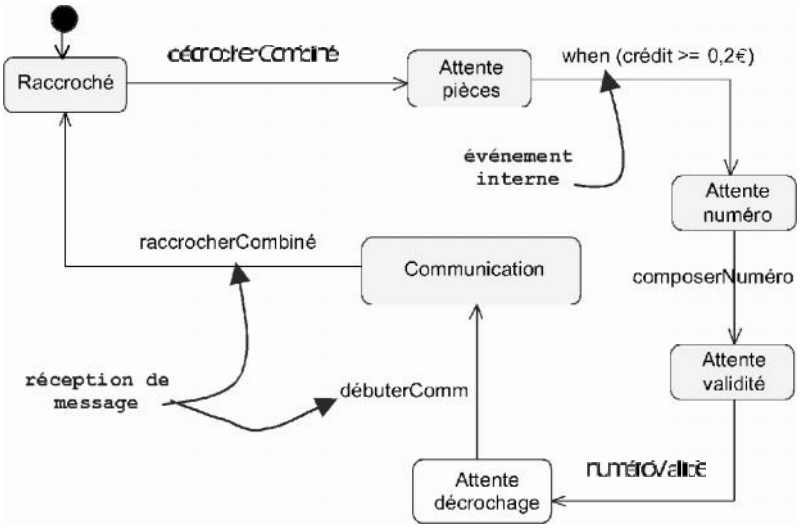
Réalisez un premier diagramme d'états qui décrive le comportement nominal du Publiphone à pièces, d'après le diagramme de séquence système.

Solution

Pour le cas d'utilisation *Téléphoner*, l'état initial nominal du Publiphone est à « raccroché en service ». Quand l'appelant décroche le combiné, il doit ensuite introduire un minimum de 0,2 € pour pouvoir composer son numéro. Une fois qu'un numéro valide est composé, le Publiphone attend la réponse du standard, puis que l'appelé décroche. La conversation est alors établie jusqu'à ce que l'un des deux raccroche. Le Publiphone revient alors à son état initial.

Traduisons ce texte en un premier squelette de diagramme d'états.

Figure 5-11.
Première version du diagramme d'états



À retenir

ÉVÉNEMENT INTERNE : « WHEN »

On notera que la plupart des événements qui déclenchent les transitions entre états correspondent à la réception d'un message émis par un acteur. Nous avons d'ailleurs utilisé les mêmes noms pour les événements que pour les messages correspondants (sauf pour *numeroValide* qui est plus lisible que le rigoureux *validiteNumero(vrai)*).

Seul le passage de l'état « Attente pièces » à l'état « Attente numéro » est provoqué par un événement interne au Publiphone : la détection du dépassement du seuil des 0,2 €. UML propose un mot-clé pour distinguer ces changements d'états internes : « `when` », suivi d'une expression booléenne dont le passage de faux à vrai déclenche la transition.

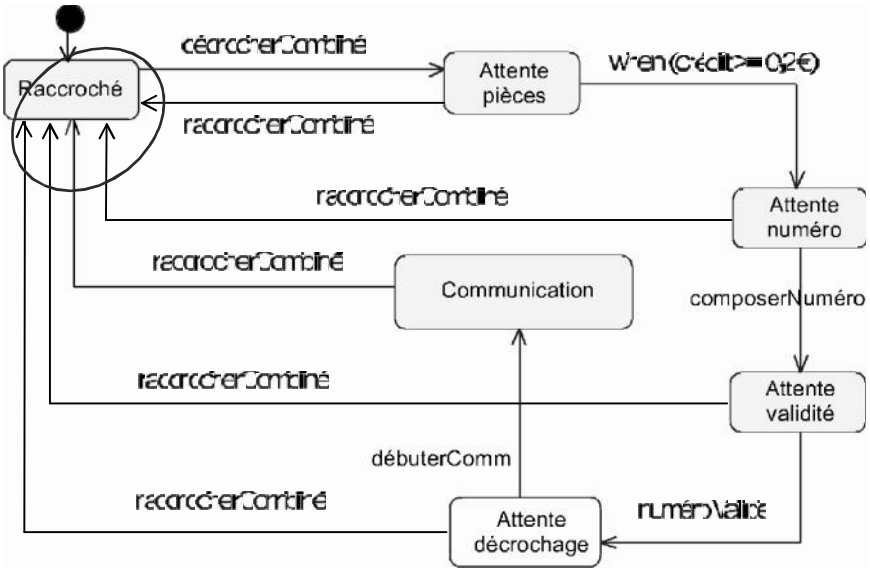
Diagramme d'états (suite)

Comment représenter le fait que l'appelant peut raccrocher à tout moment et pas seulement dans l'état conversation ?

Solution

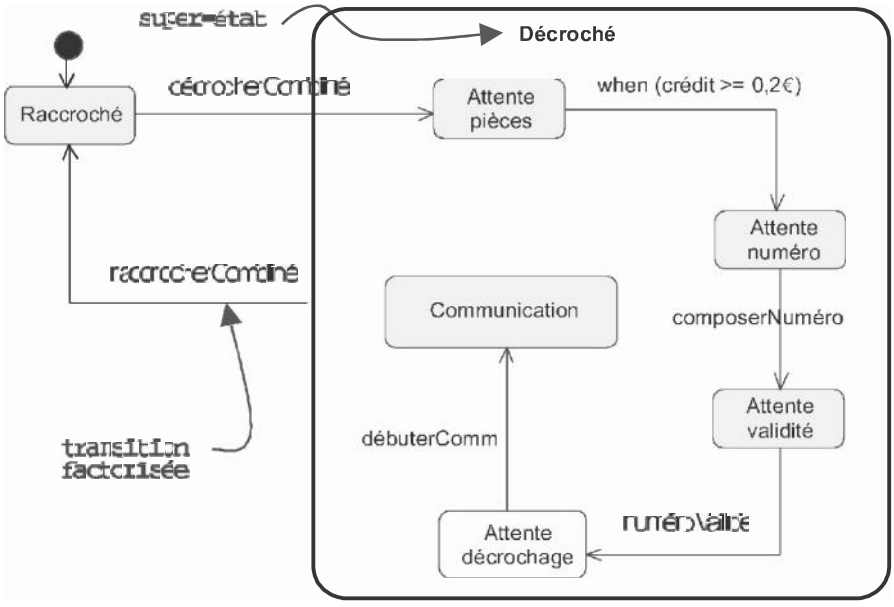
Il y a (comme souvent) deux façons de procéder : la triviale et l'élégante ! La solution triviale consiste à ajouter des transitions déclenchées par l'événement *raccrocherCombiné* et sortant de tous les états pour amener vers l'état « Raccroché ». Mais le diagramme paraît soudain bien chargé…

Figure 5-12.
Solution triviale



La solution élégante consiste à introduire un super-état, « Décroché », ce qui permet de factoriser la transition de sortie vers l'état « Raccroché ».

Figure 5-13.
Solution élégante



On notera également que nous aurions pu utiliser une notation un peu plus sophistiquée pour la transition de « Raccroché » vers « Attente pièces », comme cela est illustré sur le schéma suivant.

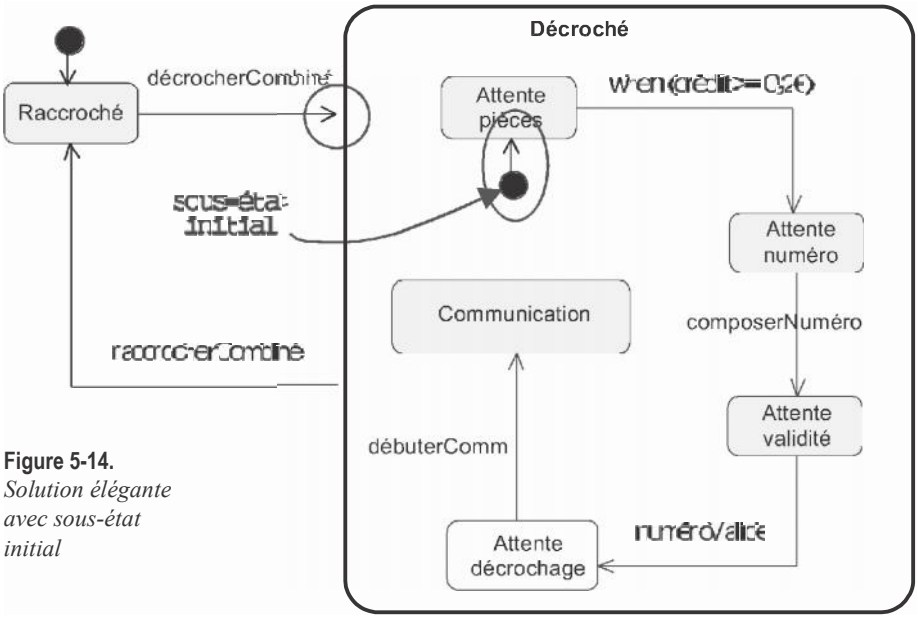


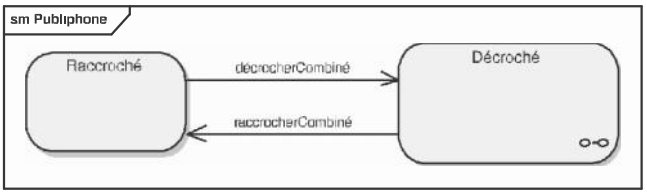
Figure 5-14.
Solution élégante avec sous-état initial

Au lieu d'avoir directement une transition entre « Raccroché » et « Attente pièces », nous obtenons une première transition entre « Raccroché » et « Décroché », puis le symbole graphique de l'état initial à l'intérieur de « Décroché », afin d'indiquer explicitement le sous-état initial. Cette manière de procéder permet de découper le diagramme d'états en deux niveaux :

- un premier niveau ne faisant apparaître que les états « Raccroché » et « Décroché » ;
- un second niveau correspondant à la décomposition de « Décroché ».

Cette possibilité de représentation sur plusieurs niveaux est illustrée sur la figure suivante, qui fait apparaître le symbole de l'état composite à l'intérieur de « Décroché ».

Figure 5-15.
Représentation des deux états de plus haut niveau[6]



Nous conserverons dans la suite de l'étude de cas la transition directe par souci de simplicité.

Diagramme d'états (suite bis)

Comment le crédit de l'appelant peut-il atteindre 0,2 € ?

Considérez plusieurs solutions.

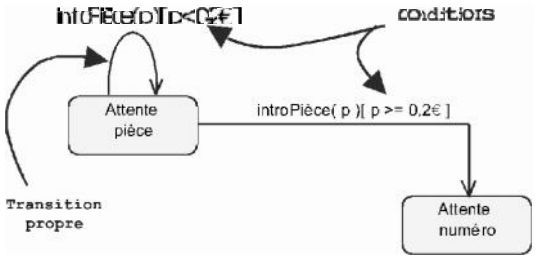
Solution

Pour le moment, le crédit de l'appelant n'intervient que dans l'expression booléenne associée à l'événement interne « `when` ». Cependant, pour que le crédit atteigne 0,2 €, il faut que l'appelant introduise une ou plusieurs pièces.

Nous pouvons donc placer une transition propre (qui ramène vers l'état de départ) sur l'état « Attente pièces ». Dès que le crédit dépasse 0,2 €, le Publiphone doit passer dans l'état « Attente numéro ».

Si nous ne souhaitons utiliser que des événements de réception de messages, probablement introduirons-nous des conditions sur les transitions comme cela est illustré sur la figure 5-16.

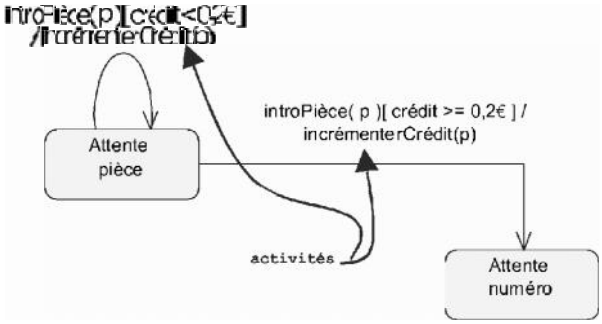
Figure 5-16.
Première solution incorrecte



Cette solution, qui paraît évidente, est cependant erronée puisqu'elle interdit de composer un numéro après avoir introduit deux pièces de 10 cents… Il faut donc que le Publiphone mémorise un attribut *crédit* qui est incrémenté à chaque introduction de pièce.

Il est tentant de transformer notre schéma en ajoutant des actions et en modifiant les conditions.

Figure 5-17.
Seconde solution incorrecte



Hélas, cette solution est également fausse, mais d'une manière plus subtile ! En effet, la sémantique d'une transition en UML est la suivante : lorsque l'événement déclencheur se produit, la condition est testée. Si (et seulement si) la condition est évaluée à vrai, la transition est déclenchée et l'effet associé est alors réalisé.

Voyons comment cela se passe concrètement sur notre exemple, en partant de « Attente pièces » avec un crédit initial de 0 € :

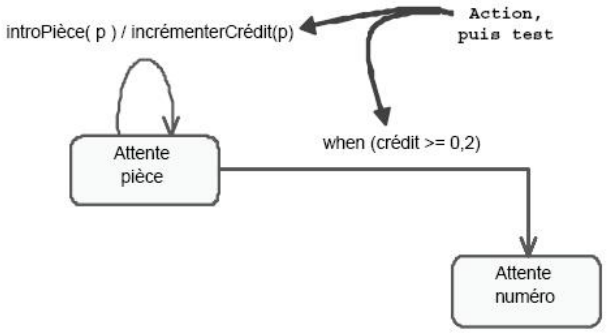
- L'appelant introduit une pièce de 10 cents. Le crédit est inférieur à 0,2 € (il vaut 0 €) : la transition propre est déclenchée et le crédit vaut maintenant 10 cents.
- L'appelant introduit une pièce de 10 cents. Le crédit est inférieur à 0,2 € (il vaut 10 cents) : la transition propre est déclenchée et le crédit vaut maintenant 0,2 €.

Le résultat est surprenant : l'appelant a payé 0,2 € et ne peut toujours pas composer son numéro… La moralité de ce constat est la suivante :

n'essayez pas de tester une donnée dans une condition avant de l'avoir modifiée par une action ou activité, ce qui est le cas lorsqu'on veut tout faire tenir dans une seule transition.

On en déduit aisément la solution correcte.

Figure 5-18.
Solution correcte



Cette façon de modéliser doit bien être notée, car elle peut être réutilisée dans nombre de contextes.

Diagramme d'états (suite ter)

Complétez la gestion du crédit de l'appelant.

N'omettez pas la dernière phrase : on peut ajouter des pièces à tout moment…

Solution

Reprenons les choses par le début. Admettons que le fait de décrocher le combiné fasse tomber les éventuelles pièces qui auraient été introduites auparavant. Le crédit doit donc être initialisé à « 0 » sur cette transition.

Ensuite, pour que le crédit atteigne 0,2 €, il faut que l'appelant introduise une ou plusieurs pièces. Nous avons donc placé une transition propre sur l'état « Attente pièces ». Dès que le crédit dépasse 0,2 €, la transition de changement « `when` » amène le Publiphone en l'état « Attente numéro ».

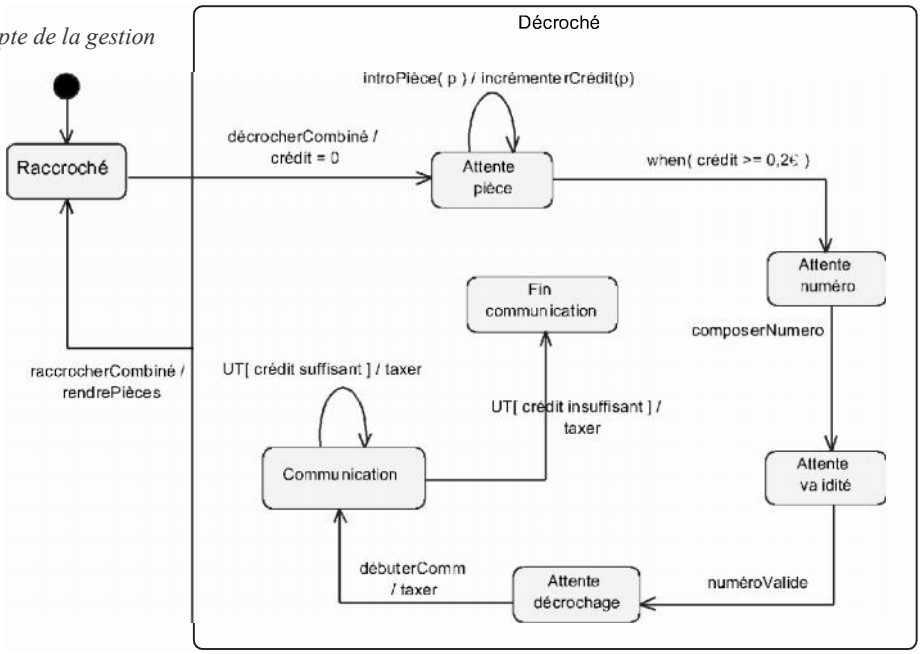
Le crédit n'évolue plus jusqu'à ce que l'appelé décroche, ce qui est traduit par le message *débuterComm* émis par le standard. En effet, à partir de ce moment là, le crédit est décrémenté régulièrement comme l'indique la phrase 5 (« Le Publiphone consomme de l'argent dès que l'appelé décroche et à chaque unité de temps [UT] générée par le standard »). L'action *taxer*

représente la chute d'une pièce à chaque impulsion.

Enfin, il ne faut pas oublier de rendre les pièces non utilisées quand l'appelant raccroche.

La prise en compte de tous ces éléments donne le schéma présenté ci-après.

Figure 5-19.
Prise en compte de la gestion du crédit



On notera l'introduction du nouvel état « Fin communication » pour parer au cas où la communication est interrompue par le Publiphone faute de crédit suffisant.

Il nous reste encore à modéliser la dernière phrase : on peut ajouter des pièces à tout moment. Là encore, plusieurs solutions sont possibles, mais une seule va s'avérer tout à la fois correcte et élégante.

La première idée consiste à introduire une transition propre identique à celle de « Attente pièce » sur chaque sous-état de « Décroché ». Cette façon de procéder est correcte, mais le dispositif est très lourd, et nous allons plutôt chercher à factoriser au moyen du super-état. N'est-il pas possible de transférer la transition propre au niveau même de « Décroché », comme cela est illustré sur le schéma ci-après ?

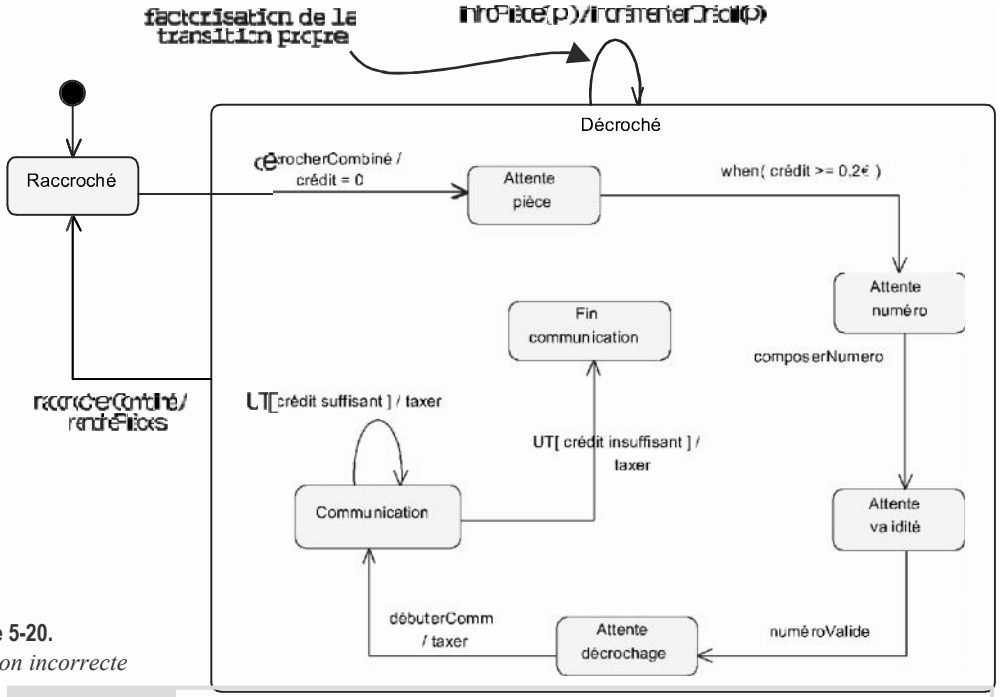


Figure 5-20.
Solution incorrecte

Hélas, cette solution n'est pas satisfaisante, comme nous allons le voir.

À retenir

TRANSITION PROPRE OU TRANSITION INTERNE ?

Dans le cas d'une transition propre, l'objet quitte son état de départ pour le retrouver. Cela peut avoir des effets secondaires non négligeables comme l'interruption puis le redémarrage d'une activité durable, la réalisation d'effets en entrée (« `entry` ») ou en sortie (« `exit` ») de l'état, etc. De plus, quand un état est décomposé en sous-états, une transition propre ramène forcément l'objet dans le sous-état initial. Ici, chaque introduction de pièce ramènerait le Publiphone dans l'état « Attente pièce », qui est implicitement sous-état initial de « Décroché » !

Pour résoudre ce problème courant, il existe en UML la notion de *transition interne* ; elle représente un couple (événement/effet) qui n'a aucune influence sur l'état courant. La transition interne est ainsi notée à l'intérieur du symbole de l'état.

C'est l'approche que nous allons utiliser pour notre étude de cas.

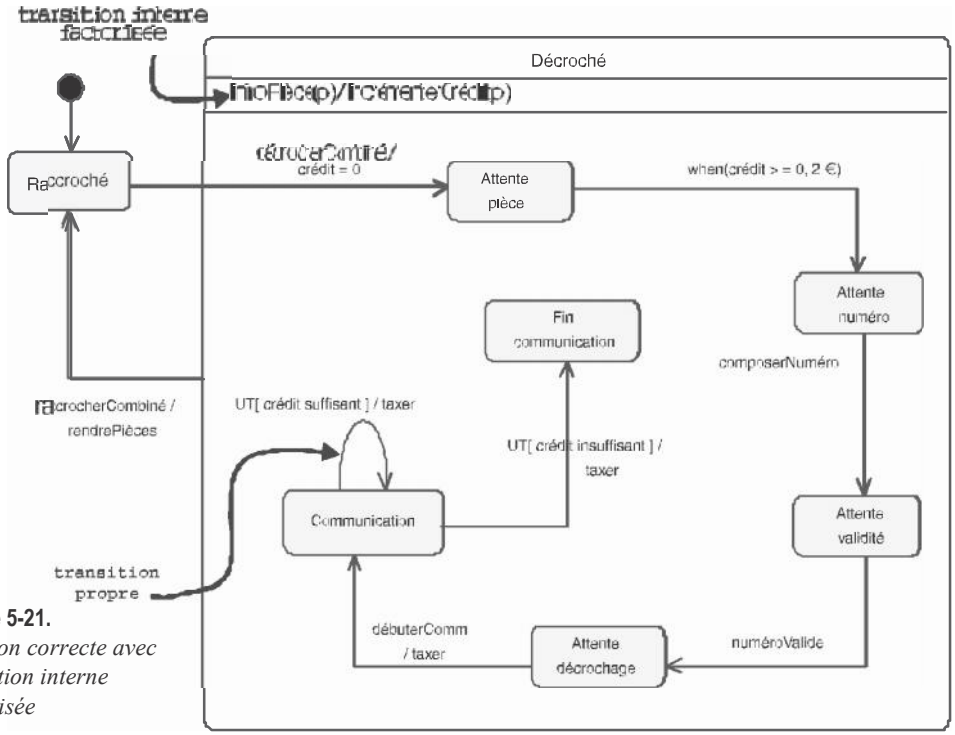


Figure 5-21.
Solution correcte avec transition interne factorisée

On voudra bien noter aussi qu'en toute rigueur, la transition propre sur l'état « Communication » devrait également être une transition interne. En fait, lorsqu'il n'y a pas d'effet secondaire, on préfère utiliser la transition propre qui est plus visuelle. Il faudra cependant être vigilant si nous devons un jour décomposer « Communication » en sous-états…

Une deuxième solution correcte, mais plus complexe, consiste à utiliser le pseudo-état « `History` ».

À retenir

PSEUDO-ÉTAT « HISTORY »

L'activation du pseudo-état « `History` » permet à un super-état de se souvenir du dernier sous-état séquentiel qui était actif avant une transition sortante. Une transition vers l'état « `History` » rend à nouveau actif le dernier sous-état actif, au lieu de ramener vers le sous-état initial.

Le diagramme 5-20 peut donc être corrigé ainsi.

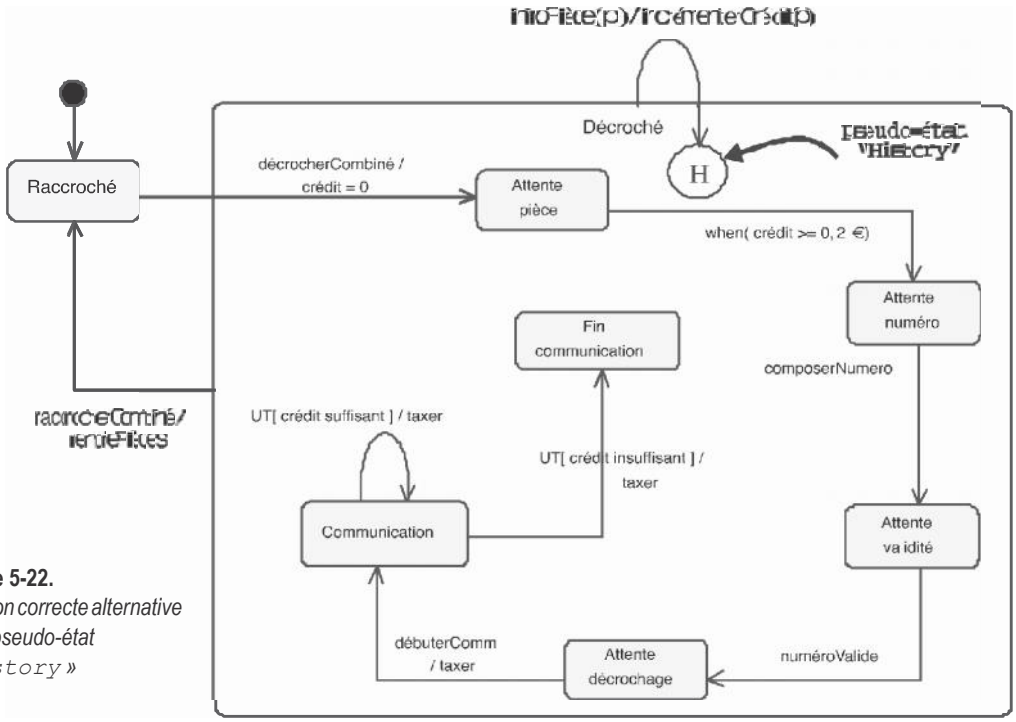


Figure 5-22.
Solution correcte alternative avec pseudo-état « History »

Diagramme d'états (fin)

Complétez le diagramme d'états pour prendre en compte l'ensemble de l'énoncé. Proposez des compléments si vous le pensez nécessaire.

Solution

Reprenons les phrases de l'énoncé une à une. Nous avons traité en détail les phrases 1, 5 et 6. En revanche, nous n'avons pris en compte que très partiellement les phrases 2, 3 et 4.

Voyons tout d'abord la phrase 2 :

2. Après l'introduction de la monnaie, l'utilisateur a 2 minutes pour composer son numéro (ce délai est décompté par le standard).

Le délai étant décompté par le standard, nous avons introduit deux messages dans le diagramme de contexte (voir figure 5-10) :

- *timerNumérotation* envoyé par le Publiphone au standard ;
- *timeoutNumérotation* envoyé par le standard au Publiphone.

À retenir

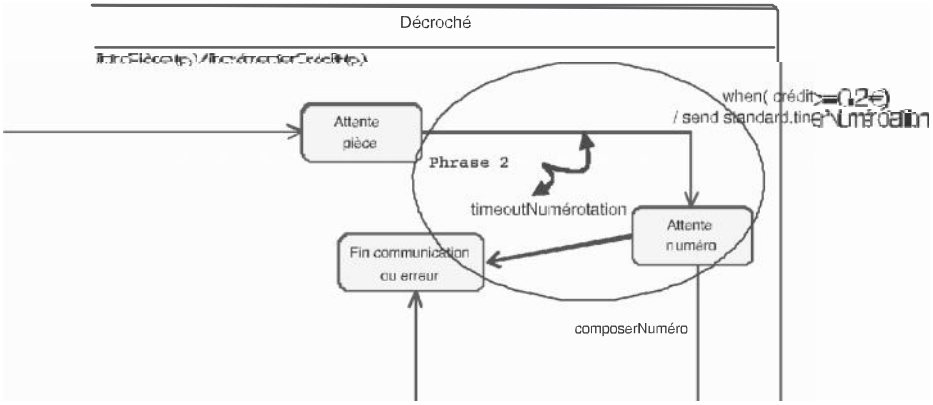
ENVOI DE MESSAGE : « SEND »

UML propose un mot-clé « `send` » pour représenter l'action importante qui consiste à envoyer un message à un autre objet sur déclenchement d'une transition.

La syntaxe de cette action particulière est la suivante : « `/ send` cible.message ».

Dans le diagramme d'états du Publiphone, nous aurons donc une transition qui sera déclenchée par la réception du message *timeoutNumérotation*, et l'envoi du message *timerNumérotation* à l'entrée de l'état « Attente numéro », comme cela est illustré sur le schéma suivant.

Figure 5-23.
Modélisation de la phrase 2



Il faut noter que nous avons renommé l'état « Fin communication », car cet « état puits » (c'est-à-dire n'ayant aucune transition en sortie) nous servira également pour tous les cas d'erreur.

Passons maintenant à la phrase 3 : la ligne peut être libre ou occupée.

Pour le moment, nous avons supposé que la ligne était libre et que le correspondant décrochait. Nous allons introduire dans le modèle la possibilité que le standard renvoie un état ligne (occupée), mais également que l'appelé ne décroche pas (ce qui n'est pas prévu explicitement dans l'énoncé). Pour ce dernier cas, nous supposons que le standard envoie un message *timeoutAppel* qui amène le Publiphone dans l'état d'erreur.

La phrase 4 ajoute simplement une transition entre les états « Communication » et « Fin communication » : le correspondant peut raccrocher le premier.

Le diagramme d'états complété est représenté ci-après sur le schéma suivant.

Figure 5-24.
Modélisation complète de toutes les phrases de l'énoncé

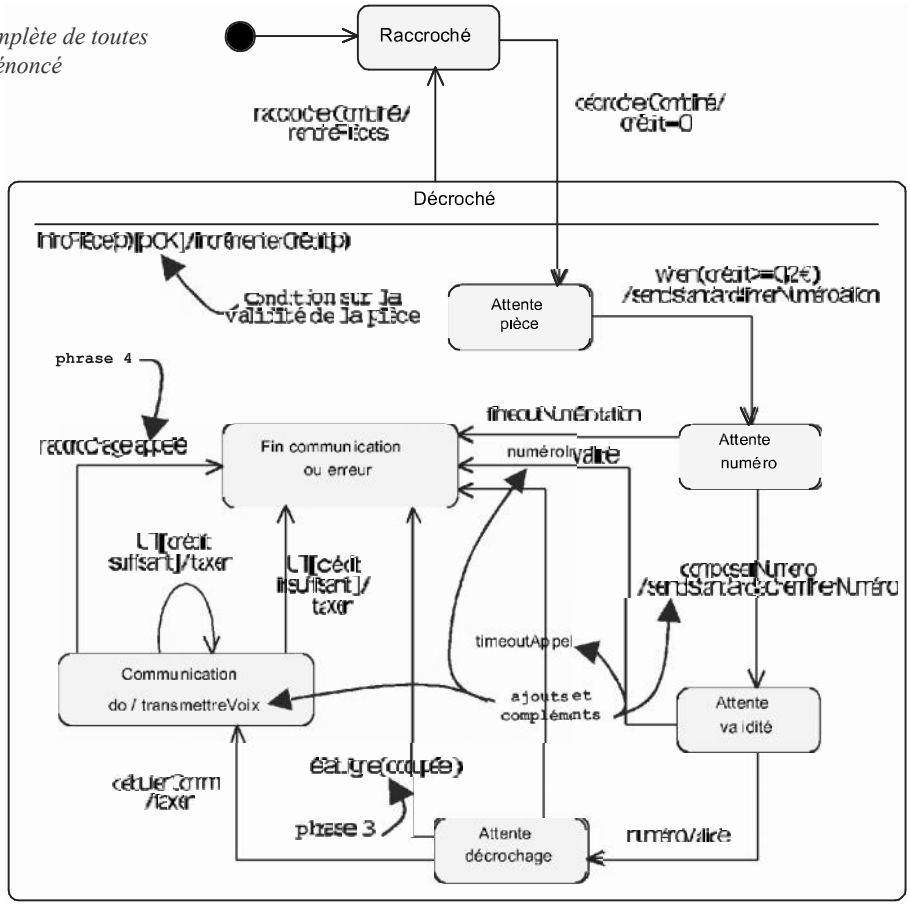


Diagramme de contexte statique étendu

En vous servant de toute cette étude dynamique, proposez une version étendue du diagramme de contexte statique qui fasse apparaître les attributs et les opérations de la classe Publiphone.

solution

Nous allons appliquer quelques règles simples :

- Les opérations publiques correspondent aux noms des messages émis par les acteurs.
- Les opérations privées correspondent aux noms des messages envoyés à soi-même.
- Les attributs correspondent aux noms des données rémanentes, manipulées dans les effets ou les conditions.

Voyons tout d'abord les opérations publiques. D'après le diagramme de contexte dynamique (voir figure 5-10), nous pouvons identifier :

- décrocherCombiné
- introPièce(p)
- composerNuméro(num)
- raccrocherCombiné
- débuterComm
- UT
- étatLigne(état)
- validitéNuméro(v)
- finComm
- timeoutNumérotation

Le diagramme d'états (voir figure 5-24) nous amène à ajouter l'opération suivante :

- timeoutAppel

Passons maintenant en revue les opérations privées. Le diagramme de séquence système enrichi (voir figure 5-8) montrait les messages suivants :

- vérifierPièce
- incrémenterCrédit
- taxer

Sur le diagramme d'états, nous avons introduit l'activité durable «do/transmettreVoix », qui peut être ajoutée à la liste des opérations privées (puisqu'elle est déclenchée indirectement par l'arrivée dans l'état « Communication »). On notera que l'opération *vérifierPièce* s'est traduite par une condition « [p OK] » sur la transition interne factorisée.

Enfin, quels sont les attributs intéressants ? Il est clair qu'une donnée importante est celle qui est gérée d'une façon permanente par le Publiphone : le *crédit* de l'appelant. Du coup, nous pouvons éliminer les opérations implicites de lecture/ écriture de cet attribut (*incrémenterCrédit*, *taxer*).

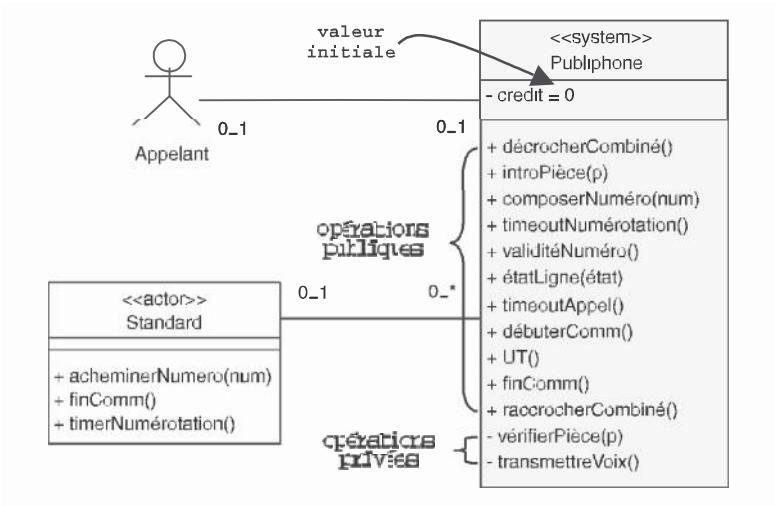
Nous en savons assez pour dessiner le diagramme de contexte statique étendu.

À retenir

DIAGRAMME DE CONTEXTE STATIQUE ÉTENDU

Nous appelons « diagramme de contexte statique étendu » un diagramme de contexte statique dans lequel nous ajoutons des attributs et des opérations de niveau système à la classe qui représente le système (appréhendé comme une boîte noire), ainsi qu'aux acteurs non- humains.

Figure 5-25.
Diagramme de contexte étendu



On notera que nous avons fait figurer les opérations publiques sur l'acteur non-humain *Standard*, mais pas sur l'acteur *Appelant*. Le concept d'opération n'a pas de sens sur un acteur humain : on ne cherche généralement pas à le modéliser d'une façon déterministe. Sur un acteur non-humain, en revanche, la liste des opérations représente son interface (au sens d'une API, par exemple) telle qu'elle est utilisée par le système étudié. C'est particulièrement utile pour vérifier l'interopérabilité des deux systèmes et s'assurer que ces opéra- tions sont déjà disponibles, ou prévues dans les spécifications.

Au point de vue notation UML, rappelons que :

- « - » signifie privé ;
- « + » signifie public ;
- « = » permet de spécifier la valeur initiale d'un attribut.