

Compléments sur les machines à états

- ❑ Elles permettent de donner une autre vision des classes : comportement dynamique d'**une** instance donnée...
 - Dans quel état une méthode est-elle appellable ?
 - Quel est son effet, dans quel état laisse-t-elle l'instance ?
 - Quelles sont toutes les méthodes qui peuvent amener une instance dans un état donné ?

Permet de vérifier qu'on oublie pas de transitions ou d'états !

- ❑ Les états peuvent rester implicites dans chaque méthode...
- ❑ ... ou au contraire, on peut programmer l'instance comme un automate
 - `switch` à la Java/C, organisé autour d'une variable qui mémorise l'état courant
- ❑ A mi-chemin entre l'analyse et la conception !

Les machines à états

- ❑ Automates perfectionnés: états, transitions + « gardes », actions
- ❑ État: abstraction de la vie d'un objet pendant laquelle il satisfait une condition, exécute certaines activités et répond à des événements extérieurs d'une certaine façon.
 - Notion d'état complexe (structuré)
 - Activités internes dans les états
 - Activités à l'entrée et à la sortie d'un état
- ❑ Transitions: relation entre deux états, reflétant généralement un changement de comportement.

Décrit par:

 - ☞ les états de départ et d'arrivée
 - ☞ l'évènement déclencheur
 - ☞ condition de franchissement (garde)
 - ☞ action(s) associée(s) à la transition

Les machines à états (2)

- ❑ Évènements : « influence » venue de l'extérieur pouvant induire une action de la part de l'instance
 - Appel (synchrone) d'une méthode de l'objet: `op (arg:T)`
 - Changement dans l'état d'une condition booléenne: `when (exp)`
 - Réception d'une communication asynchrone (signal): `sname (a:T)`
 - Écoulement d'une durée ou d'un temps absolu: `after (time)`

- ❑ Actions associées à une transition:
 - Action associée à la sortie de l'état courant (`exit`)
 - Action spécifiée sur la transition
 - Action associée à l'entrée dans l'état courant (`entry`)
 - Activité interne à l'état (`do`)

« action » : traitement atomique
« activité » : liste d'actions et/ou sous-activités



Pendant le traitement d'une transition, on ne peut pas traiter un autre évènement...

Les machines à états (3)

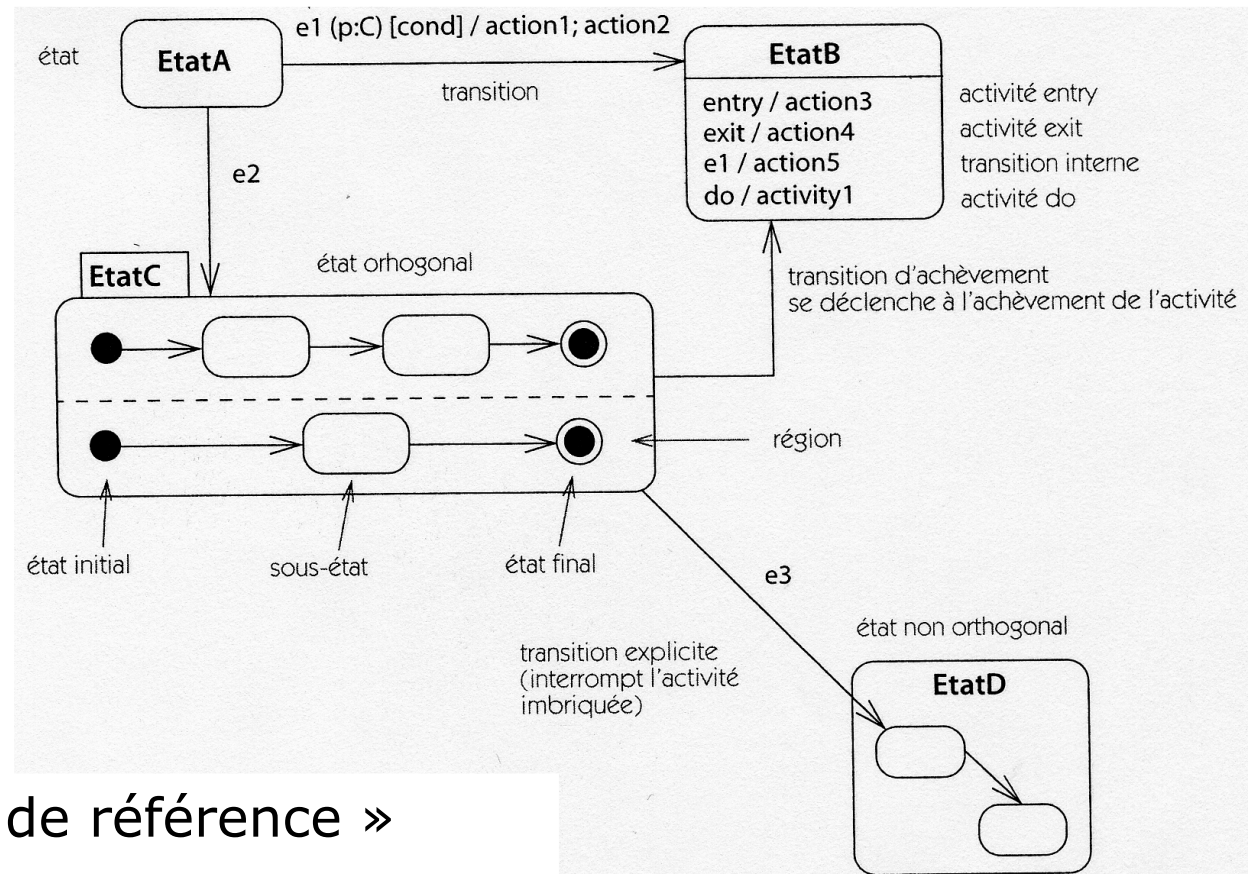
- ❑ Actions associées à une transition
 - Appel d'une méthode sur une instance
 - Modification d'un attribut
 - Envoi d'un signal à une instance
 - Création ou destruction d'un instance
 - ...

- ❑ Possibilité de « transitions internes »

- ❑ États structurés: on peut décomposer en
 - États orthogonaux (parallèles ↔ facettes indépendantes)
 - États non orthogonaux (séquence de sous-états)
 - Transitions possibles à partir de l'état global et/ou des sous-états.

☞ *Sémantique complexe, difficile à maîtriser !*

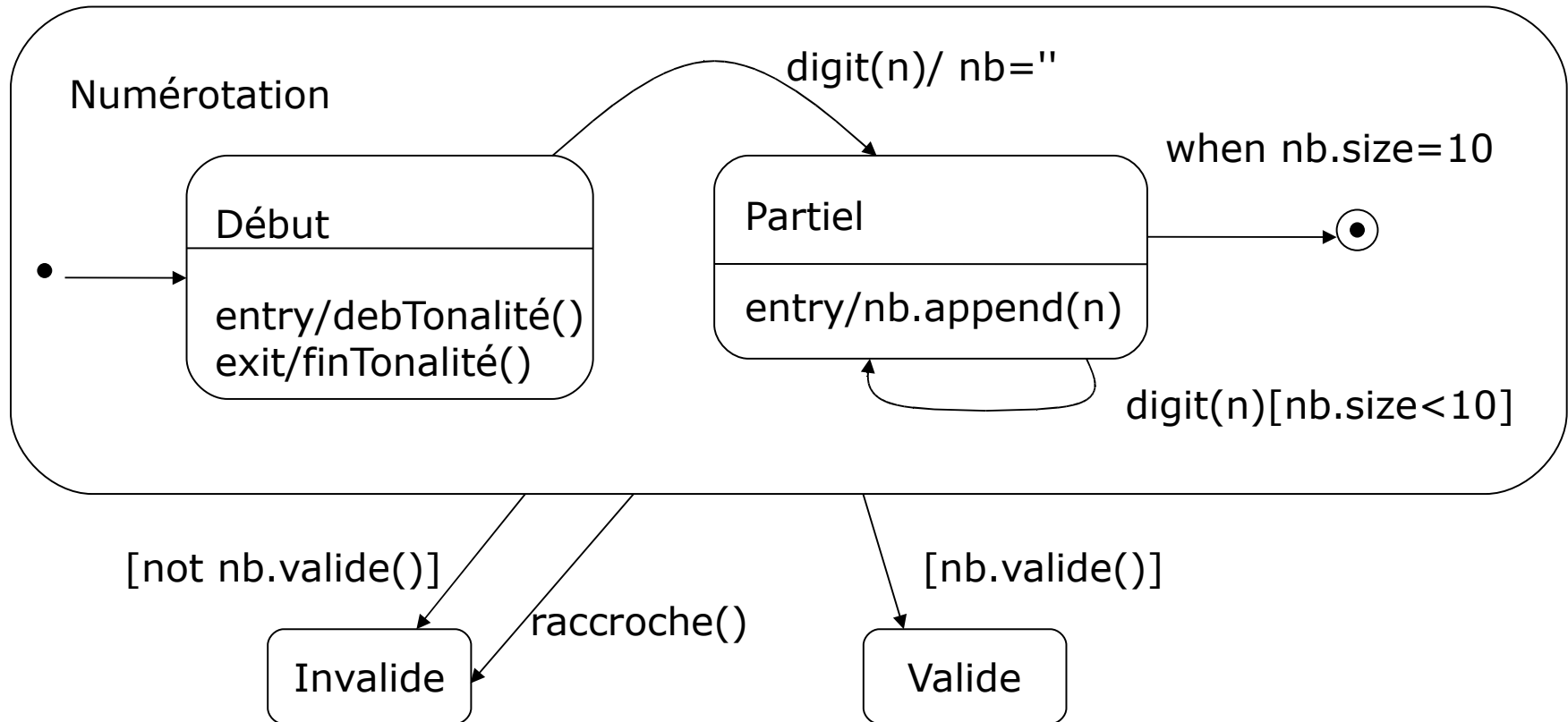
Machines à états : syntaxe étendue



« UML 2.0, Guide de référence »

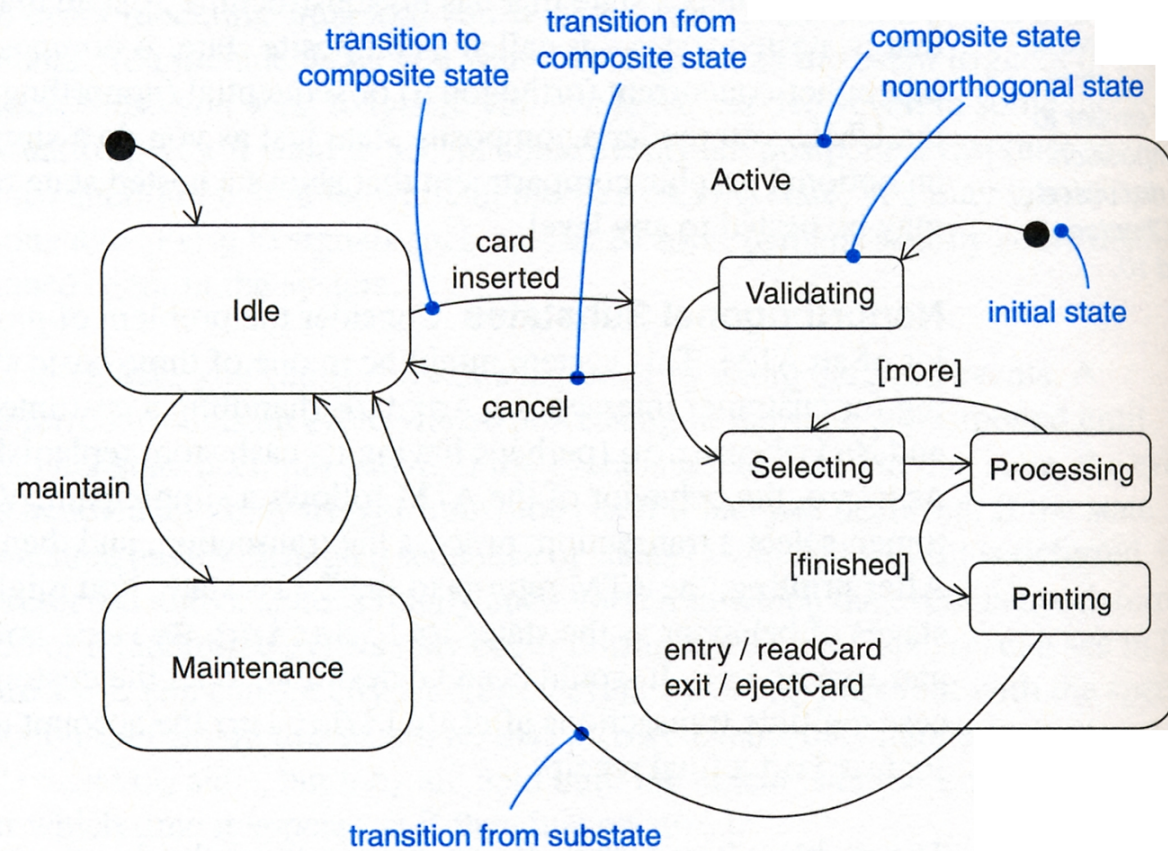
Rumbaugh & alli, CampusPress, 2005

Machines à états : exemple d'un téléphone



Exercice: modifier la machine à états pour accepter les numéros courts !

Machines à états : exemple d'un DAB



« The Unified Modeling Language User Guide »
G. Booch & alli, Addison-Wesley 2005

En parallèle avec la conception

Préparer les tests

- Déduire les tests à prévoir pour les classes:
 - ☞ Tests fonctionnels unitaires,
 - ☞ Tests d'intégration
- Tester les composants réutilisés: savez-vous vraiment comment fonctionnent les collections Java ? Tel composant graphique ? Les fonctions de l'API JDBC ?
- ☞ À tester à l'avance si c'est votre premier usage !
- ☞ Peut éviter d'avoir à chercher des bugs qui n'en sont pas, ou qui ne sont pas de votre ressort.
- Prévoir l'ordre dans lequel tester les classes !

Documents en fin de conception

- ❑ Document de conception
 - Choix de conception effectués et implications
 - Diagramme de séquence (ou autre description) pour les principales méthodes:
 - Celles implémentant des opérations des cas d'utilisation
 - Celles qui sont spécialement complexes ou qui illustrent des choix de conception
- ❑ Le dictionnaire des données mis à jour (classes, attributs, ...)
- ❑ Squelette des classes en fonction des diagrammes
 - Relations d'héritage éventuelles
 - Définition des « accesseurs » nécessaires
 - Hiérarchies d'exception à mettre en oeuvre
 - En-têtes précis pour toutes les méthodes *public/protected*
 - Identification des classes auxiliaires nécessaires
 - Décomposition en paquetages

Documents en fin de conception (2)

Document de tests (tests fonctionnels)

- A partir des descriptions UML
 - ☞ « Instantiation » des diagrammes de séquence
 - ☞ Couverture des transitions des machines à états
- Description du « contexte » pour mettre en œuvre ces test
 - ☞ Quelles instances ?
 - ☞ Dans quel état ?
 - ☞ Quels observateurs pour vérifier le résultat et l'état du système

Le tout en cohérence avec les documents d'analyse (trace des décisions et des contraintes)

Après, il n'y a plus qu'à coder, mettre au point, faire passer les tests ...