

GÉNIE LOGICIEL AVANCÉ

Cours 2 : Spécification

Yann Régis-Gianas
yrg@pps.jussieu.fr

PPS - Université Denis Diderot – Paris 7

5 février 2010

Qu'est-ce qu'une spécification ?

Des besoins du client à la spécification du système qui les satisfait

- ▶ La **spécification** établit ce que le système doit faire (le QUOI) et les contraintes sous lesquelles il doit opérer.
- ▶ L'ingénierie de la spécification consiste donc établir une **communication** entre les clients et les concepteurs du systèmes.
- ▶ Comme tout processus de communication, il s'agit donc d'un **échange d'informations** ayant pour support un **canal de communication** entre plusieurs **entités**.

- ⇒ Quelles sont ces entités ?
- ⇒ Quelle information doit être échangée ?
- ⇒ Quels sont les canaux de communication à notre disposition ?

Une étude de cas

- ▶ Le projet du cours POCA de Master 2 LP/LC/SRI :



Spécification, Conception, Développement et Maintenance
d'un *Moteur Générique de Jeux d'Aventure*

- ▶ Le format du sujet : un appel d'offre sous la forme d'un e-mail décrivant les besoins d'une société d'édition de jeux vidéos (de façon **très** informelle).

Des spécifications de natures diverses

- ▶ Il est essentiel de dissocier dans la description d'un système, les deux points de vue :
 - ▶ **interne** (celui des concepteurs, des personnels techniques, ...);
 - ▶ **externe** (celui des utilisateurs non informaticiens, des décideurs, ...).
- ▶ Pour le point de vue externe, on définit une **spécification des besoins** : une description de **haut-niveau d'abstraction des services** que doit rendre le système et les contraintes sous lesquelles il opère.
- ▶ Pour le point de vue interne, on définit une **spécification du système** : une description **la plus précise possible** du système qui doit être réalisé.

Étude de cas : spécification des besoins

Voici quelques questions qui pourraient initier l'établissement de la spécification des besoins liés à l'appel d'offre précédent :

- ▶ Pour vous, qu'est-ce qu'une aventure ?
- ▶ Pour vous, qu'est-ce qu'un scénario ?
- ▶ Quel degré d'expressivité vous serait utile ?
- ▶ Comment souhaitez-vous qu'un joueur formule ses commandes de jeu ?
- ▶ Est-ce que l'on doit pouvoir jouer à plusieurs en même temps ?
- ▶ Peut-on jouer sur Internet ?
- ▶ Peut-on jouer sur un téléphone portable ?
- ▶ Est-ce que la description d'une scène nécessite une animation ?
- ▶ Quelles compétences doivent être nécessaires à l'écriture d'un nouveau scénario ?
- ▶ ...

Nature des spécifications du système

- ▶ Il y a trois grandes catégories de spécifications de système :
- ▶ Les spécifications **fonctionnelles** : on définit les services du système en termes de relation entre les sorties et les entrées.
- ▶ Les spécifications **non fonctionnelles** : ce sont les contraintes et les propriétés remplies par le système dans son intégralité, comme, par exemple, l'efficacité, la robustesse, la sécurité, ...
- ▶ Les spécifications **liées aux domaines d'activité** : ce sont des spécifications, fonctionnelles ou non fonctionnelles, qui définissent des informations ou des contraintes liées aux règles qui régissent certains domaines.

Étude de cas : spécification du système

- ▶ Une spécification fonctionnelle du moteur générique de jeu d'aventure pourrait par exemple établir le format textuel des requêtes du joueur.
- ▶ Une spécification non fonctionnelle pourrait par exemple établir que, si le jeu d'aventure peut se jouer sur Internet, il ne bloque pas si la connexion est interrompue quelques instants.
- ▶ Une spécification liée au domaine pourrait par exemple définir précisément ce que l'on entend par « scénario ».

Différents point de vue sur les spécifications du système

- ▶ Il y a de nombreux niveaux de descriptions de la spécification du système :
- ▶ La **spécification système des utilisateurs** : c'est un **contrat** entre les concepteurs et les utilisateurs qui fixent, en des termes les plus précis possibles, ce que l'on attend du système en vue de satisfaire les besoins (qui ont été spécifiés plus tôt).
- ▶ La **spécification de l'architecture** : c'est un **contrat** entre les concepteurs et les développeurs qui établit le partitionnement modulaire du système.
- ▶ La **spécification technique** : c'est un **contrat** entre les développeurs qui établit une interface entre les composants logiciels développés indépendamment.

La vérifiabilité des spécifications non fonctionnelles

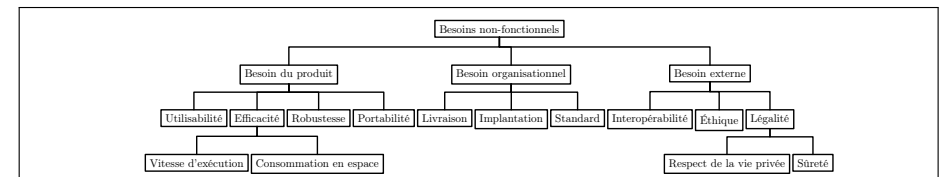
- ▶ La réalisation d'un contrat doit être **vérifiable**.
 - ▶ Or, si la spécification a été formulée en des termes imprécis, cette vérification est mise à mal.
- ⇒ Pour éviter d'interminables procès, la société moderne cherche à tout **quantifier**.
- ⇒ Il est difficile d'établir des **mesures** pour toute chose.
(exemples : un indice de divertissement, une évaluation objective du bonheur, un entier représentant la facilité d'utilisation ou l'amélioration du bien commun, la pertinence de travaux de recherche, ...)
- ▶ C'est en général la « bonne foi » de chaque partie qui est évaluée.

Étude de cas : une spécification non fonctionnelle vérifiable

- ▶ Une commande de jeu d'aventure est **simple** à effectuer si elle ne nécessite pas plus de **trois** clics de souris successifs.

Organisation des spécifications non fonctionnelles

- ▶ Une difficulté supplémentaire des spécifications non fonctionnelles est issue de leur immatérialité : on ne peut pas les associer à un composant particulier du système.
- ▶ De façon à y voir clair, on peut toutefois les organiser et les hiérarchiser de la façon suivante, par exemple :



Des spécifications du système à différents degrés de détails

- ▶ Idéalement, une spécification ne devrait se préoccuper que du **comportement observable** d'un système.
 - ▶ Cependant, le niveau de détails nécessaire pour spécifier précisément ce comportement nécessite souvent une intrusion dans la conception de ce dernier.
 - ▶ En effet, pour bien comprendre la cohérence d'un système, et en particulier, la bonne interaction entre ses composants internes ou bien les contraintes d'interopérabilité avec des systèmes externes, il faut **se placer au bon niveau de détails**.
- ⇒ La difficulté est ici de **savoir de quoi s'abstraire**.

Écrire la spécification du système à l'aide de notations

- ▶ Les langages naturels sont souvent utilisés pour spécifier les systèmes.

Tirés de l'étude d'un échantillon important de spécifications de logiciel :

<http://eprints.biblio.unitn.it/view/department/informaticas.html> (2000)

- ▶ 71.8% de ces documents sont écrits en langage naturel.
 - ▶ 15.9% de ces documents sont écrits en langage naturel structuré.
 - ▶ 5.3% de ces documents sont écrits dans un langage formalisé.
- ▶ Malheureusement, ces spécifications sont sujets à des **mécompréhensions** :
 - ▶ **Ambiguïtés** : un même mot ne fait pas toujours référence au même concept chez deux locuteurs différents ou dans deux contextes différents.
 - ▶ **Flexibilité excessive** : un même concept peut être expliqué de plusieurs façons différentes, ce qui complique la recherche d'information.
 - ▶ **Modularisation difficile** : le partage des mots entre différents modules rend difficile la tracabilité des concepts (quel module est en charge de quoi ?).

Écrire la spécification du système à l'aide de notations

- ▶ Il existe des solutions de remplacement du langage naturel libre.
- ▶ **Langage naturel structuré** : on fixe des patrons, des modèles de fiches, et des formulations, dont le sens est explicité de la façon la moins ambiguë possible dans une partie liminaire de la spécification. La suite de la spécification doit s'appuyer uniquement (ou le plus possible) sur ces constructions.
- ▶ **Description algorithmique** : un langage de programmation abstrait est utilisé pour donner une vision **opérationnelle** du système. Si la sémantique de ce langage est formalisée, alors une telle notation à l'avantage d'être formelle et pertinente en vue de l'implémentation.
- ▶ **Notation graphique** : des diagrammes, annotés généralement par du texte en langage structuré, représentent le système. Ces notations sont particulièrement pratiques pour fournir une vue d'ensemble, statique ou dynamique, d'un système ou d'un sous-système. Cependant, leur sémantique doit être précisée.
- ▶ **Spécification mathématique** : des formules ou des objets mathématiques définissent un système en s'appuyant sur des théories dont les axiomes sont explicités.

Étude de cas : une spécification en langage structuré

Titre	Vérification de la validité d'une commande du joueur
Description	Analyse la validité d'une commande en fonction de l'état de la partie.
Entrée	Une <u>commande</u> .
Précondition	Être BIEN FORMÉE.
Résultat	Un <u>booléen</u> .
Postcondition	Le résultat vaut true ssi la commande est VALIDE.

Fiche « fonctionnalité »

Titre	BIEN FORMÉE
Description	Une commande est bien formée si c'est un triplet de : ▶ une <u>action</u> DÉFINIE. ▶ un ensemble d'<u>objets</u> DÉFINIS appelé <u>acteurs</u>. ▶ un ensemble d'<u>objets</u> DÉFINIS appelé <u>cibles</u>.
Domaine	Une <u>commande</u> .

Fiche « propriété »

Étude de cas : une spécification en langage algorithmique

Soit $\mathcal{A}$: set action.

Pour tout $a \in \text{ToutesActions}$ Faire

 Pour tout $acteurs \in \mathcal{P}(\text{Objets}(\text{Scene}))$ Faire

 Pour tout $cibles \in \mathcal{P}(\text{Objets}(\text{Scene}))$ Faire

 Si $\text{Commande}(a, acteurs, cibles)$ est VALIDE Alors

$\mathcal{A} := a \cup \mathcal{A}$

 Fin Faire

 Fin Faire

Fin Faire

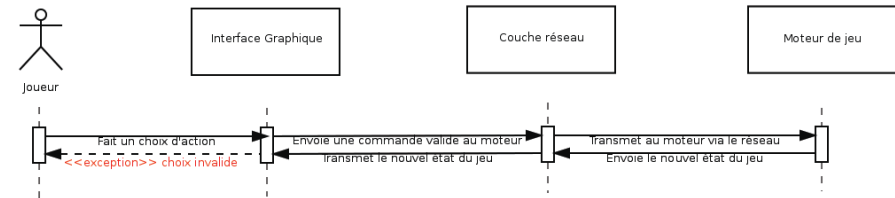
Calcul des actions possibles pour une scène fixée.

Note : cette description algorithmique serait avantageusement remplacée par l'objet mathématique

$$\{a \in \text{ToutesActions} \mid \text{VALIDE}(\text{Commande}(a, s, c)), (s, c) \in \text{Objets}(\text{Scene})^2\}$$

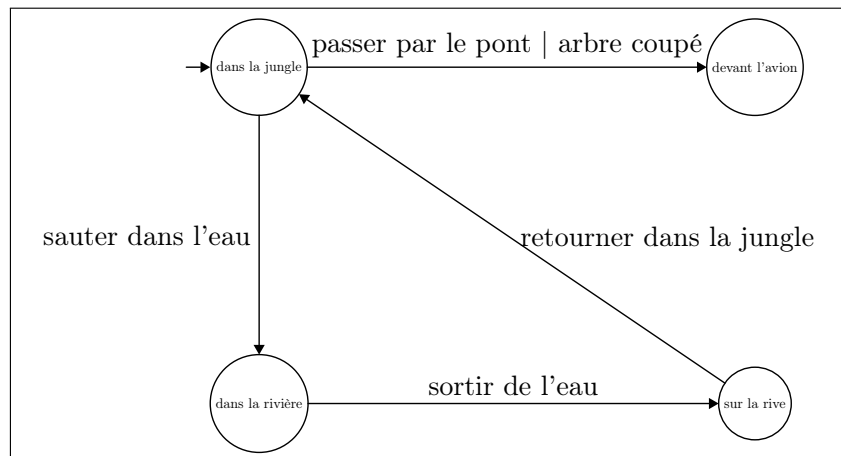
puisqu'elle n'apporte pas d'information algorithmique **pertinente**.

Étude de cas : diagramme d'interaction d'un joueur



Étude de cas : une spécification mathématique

- Un **automate de Moore** pour décrire un scénario.



Écrire un cahier des charges

- Le cahier des charges est le document **contractuel** décrivant la spécification.
- Voici un plan standard (IEEE/ANSI 830-1998) de cahier des charges :

1. Introduction

- 1.1 Objectifs du cahier des charges
- 1.2 Portée du produit
- 1.3 Glossaire : définitions, acronymes, conventions, formulations utilisées
- 1.4 Références
- 1.5 Présentation générale de l'organisation du document

2. Description générale

- 2.1 Différents points de vue sur le produit
- 2.2 Fonctionnalités du produit
- 2.3 Caractéristiques de l'utilisateur
- 2.4 Contraintes générales
- 2.5 Hypothèses et dépendences

3. Caractéristiques spécifiques

4. Annexes

5. Index

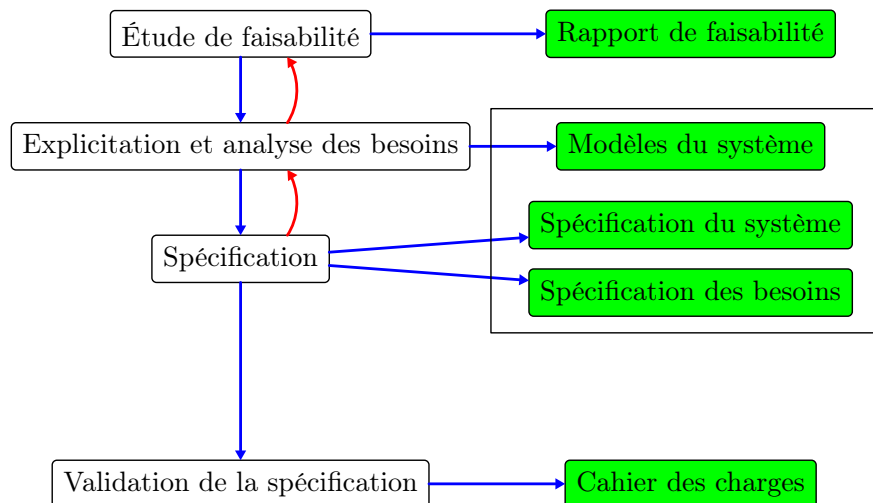
Écrire un cahier des charges

- ▶ C'est la partie 3, « caractéristiques spécifiques », qui est la plus substantielle.
- ▶ Elle est partitionnée en diverses sous-parties qui ne visent pas le même public.
- ▶ On y détaille :

Partie	Public
Résultats de l'analyse des besoins	décideurs du client, utilisateurs finaux, ingénieurs du client, gestionnaires de contrats, chefs de projet du client, architectes de système
Architecture	utilisateurs finaux, ingénieurs du client, chefs de projet du client, développeurs internes
Spécification du système	idem
Modèles du système	idem
Évolution du système	idem

Les processus de définition de spécifications

Comment définir des spécifications ?



Étude de faisabilité

- ▶ L'étude de faisabilité dresse une vue d'ensemble du rôle du système dans son éventuel environnement d'utilisation en vue d'évaluer sa **pertinence**.
- ⇒ Le rapport obtenu sert à décider si l'étude du système mérite un approfondissement ou si elle doit être suspendue.
- ▶ Cette étude est donc courte et se focalise sur les questions suivantes :
 1. Est-ce que le système contribue aux objectifs essentiels de l'environnement d'utilisation ?
 2. Est-ce que le système peut être implémenté à l'aide des technologies existantes et dans le cadre budgétaire et les contraintes temporelles fixées ?
 3. Est-ce que le système pourra être intégré aux autres systèmes déjà en place ?
- ⇒ Une **prospection** est nécessaire pour répondre à ces questions.

Étude de faisabilité

- ▶ Il faut déterminer les personnes intéressées par le système et procéder à une extraction d'informations.
- ▶ Voici des exemples de questions à poser pour ce faire :
 1. Que feriez-vous si le système n'était pas implémenté ?
 2. Quels sont les problèmes avec les systèmes existants ?
 3. Pourquoi un nouveau système améliorerait cette situation ?
 4. Quelle contribution directe apporterait un tel système sur votre travail ?
 5. Est-ce que des informations sont partagées avec des systèmes externes ?
 6. Est-ce que le système nécessiterait une technologie inconnue des utilisateurs ?
 7. Quel serait du ressort du système et qu'est-ce qu'il ne le serait pas ?

Étude de cas : différentes situations à évaluer

- ▶ Il existe déjà un moteur générique de jeu d'aventure sur le marché.
- ▶ Le temps de développement d'un jeu d'aventure spécifique est court.
- ▶ L'interaction du joueur avec le jeu d'aventure doit utiliser la voix.

Exercice

Imaginez d'autres situations qui mettraient en jeu la faisabilité du système.

Analyse des besoins

- ▶ L'analyse des besoins a lieu une fois que l'étude de faisabilité a reçu une réponse positive.
 - ▶ On doit alors **explicit** et **analyser** les besoins.
- ⇒ Il s'agit d'une **extraction minutieuse** d'information.
- ▶ Il est nécessaire de :
 1. définir les **personnes affectées** par le système ;
 2. d'évaluer leurs besoins par ordre d'**importance**.

Les personnes affectées par le système

- ▶ Les difficultés de l'interaction avec les personnes affectées par le système sont multiples :
- ▶ Elles sont pour la plupart **étrangères à l'informatique** et décrivent donc leur interaction avec le système en utilisant des termes imprécis ou exotiques. Elles n'ont pas de notion de ce qui est réalisable ou non.
- ▶ Elles n'ont pas conscience de la **connaissance implicite** sur leur travail qu'elle véhicule dans leur discours. Pourtant, c'est exactement la connaissance que vous devez extraire !
- ▶ Les différentes catégories d'utilisateur ont des besoins et des points de vue différents sur le système et la hiérarchisation des fonctionnalités.
- ▶ Des considérations politiques peuvent rentrer en jeu.
- ▶ L'environnement économique, par définition changeant, joue un rôle important sur l'évolution possible du système.

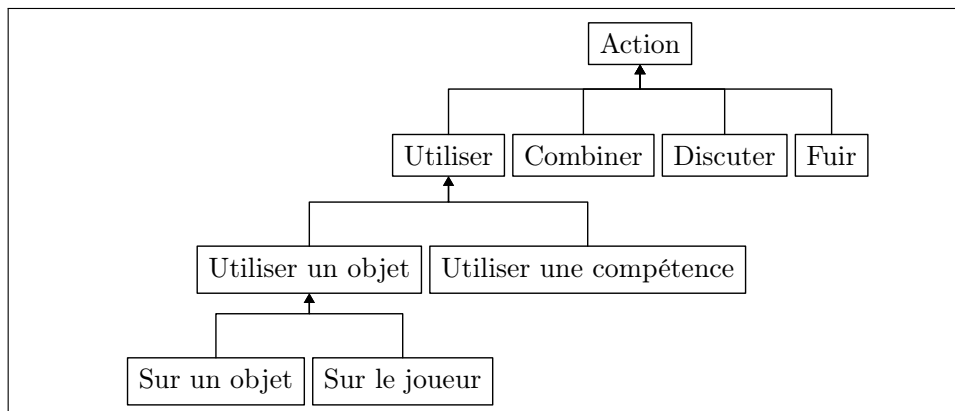
L'évaluation hiérarchisée des besoins

- ▶ Il est essentiel d'expliciter les besoins les plus importants en premier lieu.
- ⇒ Dans le cas contraire, on prend le **risque** de manquer une fonctionnalité importante et de reprendre à zéro sa compréhension du système dans son ensemble.
- ▶ On peut décomposer cette étape en sous-étapes :
 1. Explicitation des besoins ;
 2. Classification et organisation des besoins ;
 3. Définition des priorités et négociation ;
 4. Documentation des besoins.

Explicitation des besoins

- ▶ Il faut définir ses interlocuteurs.
- ⇒ Chacun d'eux définit un **point de vue** sur le système.
- ▶ On peut classer ses points de vue en trois catégories :
 - ▶ Celui des personnes qui interagissent avec le système ;
 - ▶ Celui des personnes qui ont une relation indirecte avec le système ;
 - ▶ Celui, plus général, lié au domaine d'application.
- ▶ En fonction de ses points de vue, différentes formes de besoins apparaissent.
- ▶ Pour classer à nouveau ces « sous-points de vue », on peut essayer d'identifier des points de vue plus spécifiques comme ceux :
 - ▶ Des fournisseurs de services et de ceux les utilisant ;
 - ▶ Des systèmes devant s'interfacer avec le système à spécifier ;
 - ▶ Des règles et des lois régissant le cadre d'utilisation du système ;
 - ▶ Des ingénieurs qui développeront et s'occuperont de la maintenance ;
 - ▶ Du marketing et des autres points de vue liés à l'image du système.

Étude de cas : les points de vue sur le moteur générique



Extraction de connaissances

- ▶ L'extraction de connaissance est un exercice difficile qui doit être préparé.
- ▶ On peut préparer un ensemble de questions précises.
- ▶ On doit rester ouvert d'esprit et être à l'écoute de son interlocuteur, sans préjugés.
- ▶ Il faut faire preuve de qualité de communication.
- ▶ Par exemple, plutôt que de demander à quelqu'un : « Que voulez-vous ? », il est préférable de poser des problèmes concrets.
- ⇒ En général, il est plus facile d'engager une discussion sur un sujet concret que sur des notions abstraites.

Utilisation de scénario

- Utiliser une session d'interaction comme support de la discussion facilite la projection de votre interlocuteur comme utilisateur du futur système.
- Un scénario peut inclure :
 1. Une description du contexte dans lequel le scénario démarre.
 2. Une description du flot normal des événements d'interaction.
 3. Une description de ce qui peut mal se dérouler (erreur, panne, incomplétude, ...) et de la façon dont se passe la résolution du problème.
 4. Des informations sur les activités qui peuvent se dérouler de façon concurrente.
 5. Une description de l'état du système à la fin du scénario.

Étude de cas

Contexte	Une partie est en cours. Le joueur a formulé une requête d'action.
Flot normal	La requête est exaucée. L'état de la partie est modifiée en accord avec le scénario et l'interface graphique est mise-à-jour. Un message textuel informe le joueur du changement.
Cas problématique	L'action n'est pas applicable. Le joueur est informé des causes de l'erreur. Il peut formuler une autre action.
Activités concurrentes	Les animations de la scène se poursuivent tout au long de la résolution de la requête d'action.

Scénario « résolution d'une action »

Cas d'utilisation

- Les cas d'utilisation (*use-cases*) sont une technique fondée sur les scénarios qui jouent un rôle central dans le *Rational Unified Process* que nous étudierons au prochain cours.
- C'est un des piliers de la notation UML utilisée pour spécifier les systèmes à l'aide d'un modèle à objets.
- *Stricto sensu*, un cas d'utilisation regroupe plusieurs scénarios que l'on regarde à un niveau d'abstraction suffisamment haut pour les assimiler.
- Un diagramme de cas d'utilisation en UML explicite :
 - Les **acteurs** du système, c'est-à-dire les entités qui interagissent avec lui et lui sont **extérieurs**.
 - Les scénarios du système, regroupés et organisés suivant trois relations principales :
 - la généralisation : explicite des sous-ensembles de scénario ;
⇒ « Faire X, c'est une façon particulière de faire Y » ≡ « Y généralise X »
 - l'inclusion : explicite un préfixe d'un scénario ;
⇒ « Pour faire X, il faut d'abord faire Y » ≡ « Y est inclus dans X »
 - l'extension : explicite des événements supplémentaires.
⇒ « Faire Y, c'est faire X et Z » ≡ « Y étend X »

Ethnographie

- Les systèmes sont utilisés dans un environnement « réel ».
 - Par nature, les pratiques sont différentes des abstractions théoriques.
 - Il se peut qu'un interlocuteur est un discours sur son travail différent de sa réalité quotidienne.
- ⇒ Une **immersion** dans la réalité de l'environnement d'utilisation du système renseigne parfois beaucoup plus que des discussions organisées pour déterminer comment ses interlocuteurs travaillent *réellement*.

Étude de cas

- ▶ Observer comment un joueur de jeu d'aventure interagit avec l'interface.
- ▶ Découvrir les processus de création des scénaristes et graphistes.
- ▶ S'intéresser aux communications inter-joueurs dans un jeu d'aventure.
- ▶ ...

Validation de la spécification

- ▶ La **validation de la spécification** vise à montrer que les besoins explicités correspondent effectivement à ceux des utilisateurs du système.
- ▶ Elle sert aussi à débusquer les erreurs dans le cahier des charges :
« Plus tôt une erreur est détectée et moins elle coûte. »
- ▶ En effet, détecter une erreur dans la spécification une fois que la conception, le développement et la validation du système sont terminés impliquent une nouvelle passe sur toutes ses phases !

Points de vérification

- ▶ Vérification de la validité : une fois ses besoins explicités, il se peut qu'un acteur revienne sur ses déclarations et réexprime ses besoins différemment.
- ▶ Vérification de la cohérence : les besoins exprimés dans le cahier des charges ne doivent pas être contradictoires.
- ▶ Vérification de la complétude : le cahier des charges doit, tant que possible, contenir la totalité des besoins des acteurs.
- ▶ Vérification du réalisme : on doit vérifier que les besoins peuvent être effectivement satisfaits à l'aide de la technologie existante et en respectant le budget et les délais.
- ▶ Vérifiabilité : on doit s'assurer que les besoins sont exprimés sous une forme vérifiable, avec le moins d'ambiguïtés possibles, de façon à ce que le document puisse faire office de contrat.

Procédés de vérification

- ▶ Relecture systématique du cahier des charges par un tiers.
- ▶ Écriture d'un prototype. (cf. les processus évolutifs)
- ▶ Génération de cas d'utilisation exécutables. (cf. *Extreme Programming*)

Développement d'une spécification

- ▶ Il n'est pas rare que l'écriture d'un cahier des charges nécessite plusieurs itérations.
- ⇒ Définir une spécification est un **développement** en soi.
- ▶ Garder une trace de l'évolution d'une spécification est essentiel car elle peut servir à justifier les choix qui ont été pris : on évite de revenir en arrière dans le développement.
- ▶ Maintenir une dépendance entre les différentes parties de la spécification permet de déterminer rapidement les parties impactées par une modification ou la prise en compte d'une meilleure compréhension d'un point donné.
- ▶ Lorsqu'une modification de la modification est issue d'une évolution du logiciel, on peut évaluer les coûts et les conséquences en termes de modifications de l'implémentation plus efficacement.

Comment modéliser un logiciel ?

Un point de vue technique sur le logiciel

- ▶ La modélisation du logiciel est l'activité la plus **scientifique** de la rédaction du cahier des charges.
- ▶ En effet, bien qu'on se cantonne encore au domaine du « QUOI ? », les modèles logiques utilisés par la spécification doivent être le plus précis et le moins ambigu possible.
- ▶ On applique ici, de façon intensive, le **principe de séparation des problèmes en sous-problèmes** et ses instanciations que sont l'**abstraction** et la **modularité**.
- ▶ Il y a encore différents points de vue à adopter :
 - ▶ le point de vue **externe** qui modélise l'environnement et le contexte d'utilisation du système ;
 - ▶ le point de vue **comportemental** qui décrit le fonctionnement observable du système ;
 - ▶ le point de vue **structurel** qui s'intéresse à l'architecture du système et des données qu'il traite.

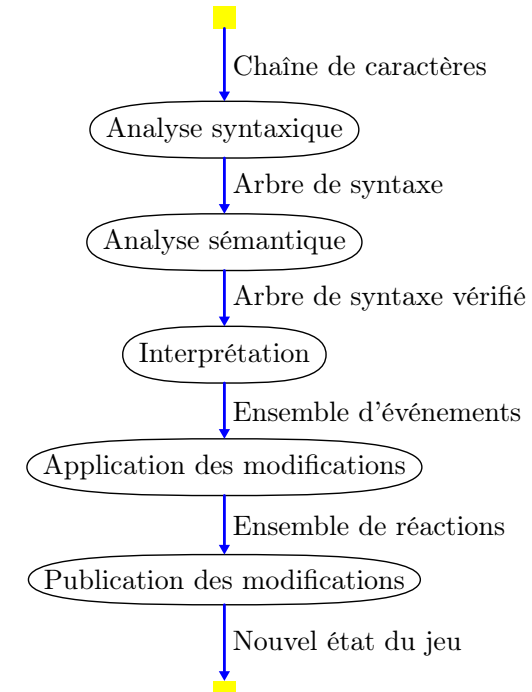
Exemples de modèles

- ▶ Modèle de **flot de données** :
Comment les données sont traitées à chaque niveau du système.
- ▶ Modèle de **composition** :
Quelles sont les entités du système et comment elles interagissent.
- ▶ Modèle **architectural** :
Quels sont les principaux sous-systèmes.
- ▶ Modèle de **classification** :
Quelles sont les caractéristiques communes aux composants du système.
- ▶ Modèle **réactif** :
Comment le système réagit aux événements internes ou externes.

Modèle contextuel

- ▶ Un modèle est **contextuel** lorsqu'il sert à déterminer la **frontière** du système.
- ▶ La définition de ce type de modèle est nécessaire à la poursuite de l'analyse puisqu'elle détermine son sujet.
- ▶ Exemples de modèles contextuels :
 - ▶ les modèles architecturaux.
 - ▶ les **modèles de processus** : on décrit les activités de l'utilisateur comme un processus et on exhibe le sous-ensemble d'activité dont est responsable le système.

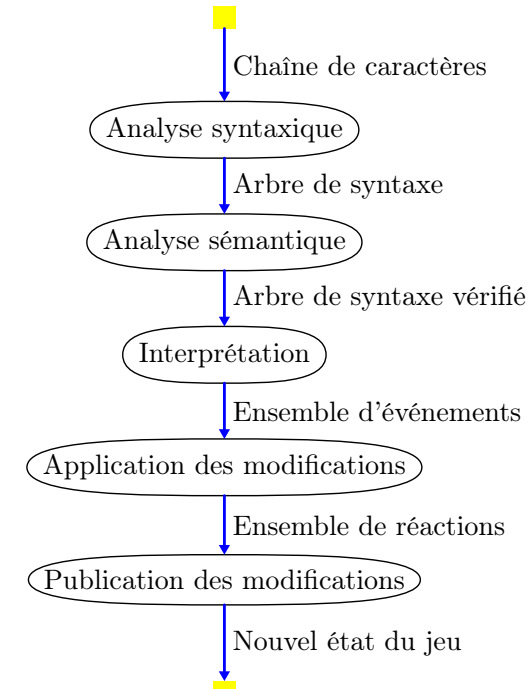
Étude de cas : la frontière du moteur de jeux d'aventure



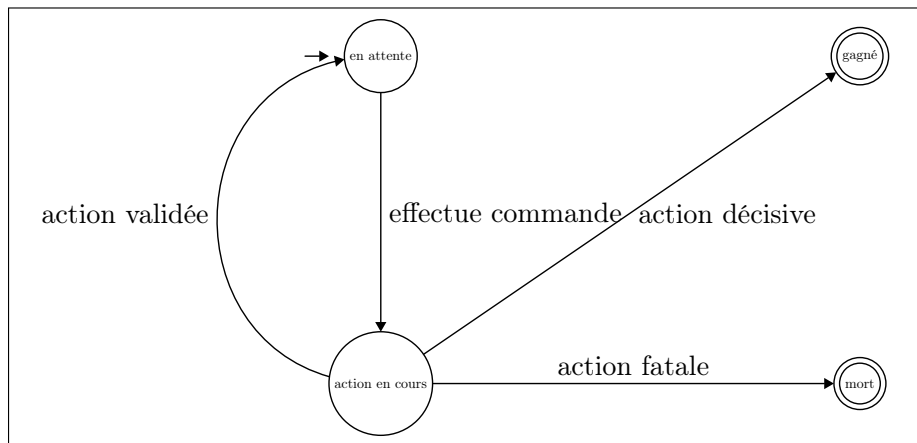
Modèle comportemental

- ▶ Les modèles comportementaux décrivent le fonctionnement observable du système.
- ▶ Exemples :
 - ▶ Modèles de flot de données : les composants logiciels sont vus comme des **fonctions** qui transforment des données d'entrée en données de sortie.
 - ▶ Modèles de machine à états finis : le système est représenté par un **transformateur de signaux**. Un signal est une succession d'événements (appelée aussi *stimulis* en entrée et *réactions* en sortie.) Les états du système sont énumérés et les transitions entre ces états sont conditionnées par les événements et éventuellement par des contraintes exprimées à l'aide de formule logique ou d'une information codant des propriétés calculatoires externes (jetons pour modéliser des ressources, points de rendez-vous, ...).

Étude de cas : interprétation d'une commande du joueur



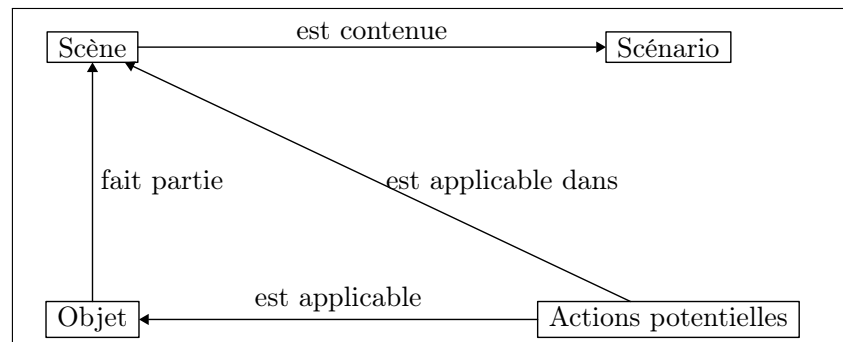
Étude de cas : état du joueur dans une partie



Modèle orienté données

- ▶ Les systèmes informatiques manipulent souvent des bases de données.
- ▶ Les modèles orientés données se concentrent sur l'organisation de ces dernières.
- ▶ Pour cela, on utilise des modèles **relationnels** ou **entité-relation**.
- ▶ Ils définissent les différentes catégories de données et les relations (statiques) qui contraignent leur agencement.

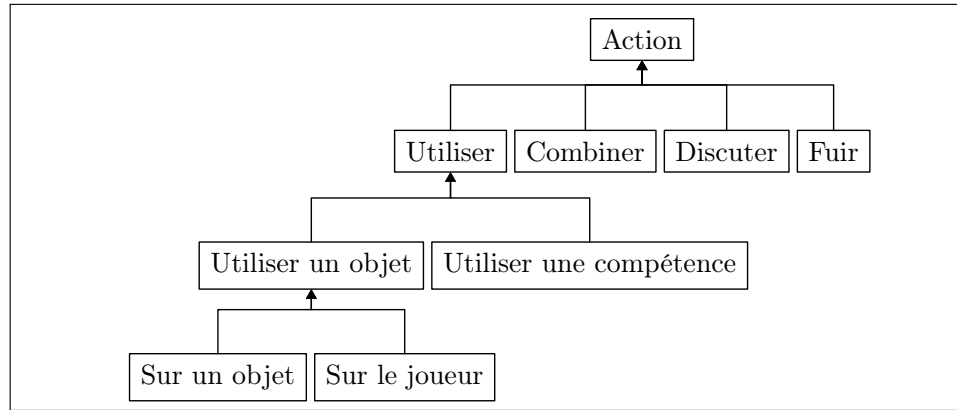
Étude de cas : Données de description d'un scénario



Modèle à objets

- ▶ L'approche orientée objet modélise un système sous la forme d'un ensemble d'objets **possédant un état propre** et interagissant par le biais d'**envoi de messages**.
 - ▶ C'est la plus courante dans l'industrie du logiciel contemporaine.
- ⇒ La spécification d'un modèle à objets sera l'objet du prochain cours.
- ⇒ Nous étudierons la notation semi-formelle UML.

Étude de cas : Taxonomie des actions d'un joueur



Synthèse

Écrire une spécification

- ▶ La notion de **spécification** recouvre différents aspects fonction du point de vue adopté sur le système.
- ▶ Définir une spécification, c'est expliciter des besoins de la façon la plus méthodique, complète et cohérente possible.
- ▶ Une spécification se développe, se vérifie, et évolue.
- ▶ Elle se concrétise sous la forme d'un cahier des charges qui dirige la conception du système et sert de contrat entre les différentes parties.

QCM : Question 1

La spécification des besoins du client doit mentionner :

- ☐ le langage de programmation à utiliser.
- ☐ les choix des structures de données.
- ☐ la place du système dans son activité.
- ☐ l'organisation de la base de données du système.

QCM : Question 2

Les propriétés suivantes font partie de la spécification fonctionnelle d'un programme de tri de tableau :

- ☐ les éléments du tableau sont préservés.
- ☐ les éléments du tableau sont triés.
- ☐ l'algorithme s'exécute en temps $O(n.\log(n))$.
- ☐ l'algorithme fonctionne sur des tableaux de dimension arbitraire.

QCM : Question 3

Les propriétés suivantes font partie de la spécification non fonctionnelle d'un compilateur :

- ☐ la sémantique du programme compilé est la même que celle du langage source.
- ☐ le compilateur rejette les programmes mal formés.
- ☐ le compilateur respecte les recommandations d'Andrew Appel.
- ☐ le compilateur produit des messages d'erreur en français.

QCM : Question 4

Les spécifications suivantes sont vérifiables :

- ☐ la quantité de mémoire consommée n'excède pas 1Mo.
- ☐ les effets spéciaux du jeu sont impressionnants.
- ☐ le programme termine toujours en moins de 10ms.
- ☐ le serveur supporte la montée en charge.
- ☐ l'interface graphique est ergonomique.

QCM : Question 5

Les besoins de différents utilisateurs :

- ☐ doivent tous être explicités.
- ☐ peuvent être contradictoires.
- ☐ sont aisément définissables en discutant avec eux.
- ☐ sont clairs dans leur esprit.
- ☐ n'ont pas le droit d'évoluer.

QCM : Question 6

Un modèle du système sert à :

- ☐ pouvoir en faire des copies à l'infini.
- ☐ le décrire selon un certain point de vue abstrait.
- ☐ fournir un cadre clair facilitant sa réalisation.
- ☐ déceler des incohérences dans l'analyse de besoin.
- ☐ valider le système dans son ensemble.