Chapitre 2

Introduction et étude préliminaire

2.1 Présentation de la méthode de Pascal Roques

Nous allons examiner les étapes préliminaires de la méthode de Pascal ROQUES telle qu'il l'expose de manière empirique dans son ouvrage *UML, Modéliser un site e-commerce* et dans un ouvrage plus théorique, *UML en action* [1]. Afin de respecter les droits d'auteur, tout en profitant de l'intérêt de la méthodologie présentée, je me contenterai de quelques allusions au cas pratique de Roques, me réservant d'en exposer les détails oralement en classe et laissant aux étudiants le loisir de se tourner vers les ouvrages de Roques pour y trouver les études de cas originales.

La méthode 2TUP (2 Track Unified Process) reprend, comme il a été dit dans l'introduction, les principes de la méthode RUP. Quatre phases interviennent successivement dans la gestion d'un tel processus :

- la *préétude* envisage les fonctionnalités du futur système logiciel (et pose également la question de sa faisabilité)
- l'*élaboration* développe l'architecture technique et réalise les fonctions les plus prioritaires
- la *construction* livre progressivement toutes les fonctions du système
- la *transition* permet de corriger et de faire évoluer le système.

Processus itératif et incrémental

La nécessité de produire de l'exécutable à intervalles réguliers permet d'éviter les risques d'échec. Roques considère un délai supérieur à 9 mois comme annonciateur d'échec du projet. A chaque itération, on peut vérifier que l'équipe garde la maîtrise de son environnement technique. L'utilisateur a également l'impression de toucher du concret : les séances d'analyse débouche sur un outil utilisable, ce qui le motive à continuer à fournir son aide pour le développement de la suite et pour l'évaluation des parties réalisées.

L'incrémentation peut porter sur une meilleur évaluation du projet, une vérification de l'adéquation aux besoins, un ajout de fonctionnalités ou une correction/évolution selon la phase où elle se situe.

1. Pascal ROQUES, *UML, Modéliser un site e-commerce*. Eyrolles, collection les Cahiers du Programmeur, 2002. Pascal ROQUES et Franck VALLÉE, *UML en action, De l'analyse des besoins à la conception en Java.* Eyrolles, 2003. Les ouvrages ont été réédités plusieurs fois avec des modifications mineures.

Processus piloté par les risques

La prise en compte précoce des risques permet de prendre des mesures efficaces qui ne consistent pas en des mises à jour correctrices de dernière minute. Par exemple, dans un système où la réactivité est primordiale, on ne se contentera pas de créer un programme quelconque puis de l'optimiser. On choisira des techniques et des outils dont on connaît à l'avance la rapidité. Dans un contexte où la sécurité prime, l'ensemble de l'architecture sera construite en prenant la sécurité comme axe central.

Création et maintenance d'un modèle

Plutôt que des documentations éparses, susceptibles de devenir obsolètes, on gère un modèle avec différents niveaux d'abstraction et niveaux de détails. On privilégie la notation UML. Cela ne dispense pas d'une organisation stricte, surtout pour des projets impliquant un grand nombre de participants.

Processus orienté composant

Le développement avec des technologies complexes, faisant appel à des progiciels du marché, des composants de diverses natures, contraint les analystes à soigner la structuration des différents composants, qu'ils soient développés expressément, simplement importés dans le projet ou qu'ils assurent l'interface avec des composants extérieurs. On accorde une attention particulière à la notion de paquetage (*package*) qui regroupe des composants qui assurent des fonctionnalités réutilisables dans d'autres contextes.

Processus orienté utilisateur

Cette perspective s'envisage doublement : on part des cas d'utilisation, dûment recensés. En outre, on établit des priorités, ce qui permet de consacrer les premières itérations à la réalisation des fonctionnalités les plus urgentes et les plus attendues.

Processus en forme de Y

Caractéristique originale de la méthode de Roques, la distinction d'une branche fonctionnelle et d'une branche technique offre de nombreux avantages :

- on prend en compte des contraintes et des exigences qui portent d'une part sur le métier des utilisateurs (branche fonctionnelle) et sur l'environnement (branche technique).
- cette distinction permet la réutilisation ultérieure : soit reprendre la même analyse pour un contexte différent (changement de technologie), soit profiter d'une expertise technique pour l'adapter à des fonctions différentes.

Différentes étapes peuvent s'observer.

Branche gauche (fonctionnelle)

1. capture des besoins fonctionnels : focalisation sur le métier et sur les utilisateurs
2. analyse : production d'une description du système, indépendante de toute technologie particulière

Branche droite (architecture technique)

1. capture des besoins techniques : examen des contraintes et des choix techniques, prise en compte de l'existant
2. conception générique : définition des composants, uniformisation, réutilisation. On gère les risques de niveau technique.

Branche du milieu

1. conception préliminaire : intégration du modèle d'analyse dans l'architecture technique, prévision des composants du système à développer
2. conception détaillée : on envisage la manière de réaliser chaque composant
3. codage et tests : réalisation effective et test de chacun des composants logiciels
4. recette et validation : livraison et validation du système

La fin du chapitre va envisager plus en détails l'étape initiale du processus.

2.2 Capture initiale des besoins (2TUP)

La capture initiale des besoins se réalise au moyen d'entrevues avec les responsables de la société. Elle se découpe en deux parties :

- recueil initial des besoins fonctionnels et technique qu'on résume dans un premier document : le cahier des charges
- modélisation du contexte

2.2.1 Cahier des charges

Il s'agit d'un document écrit, sous forme textuelle, qui fait la synthèse des observations réalisées dans l'entreprise.

Présentation du projet

Il s'agit ici de présenter la société ou le service concerné par l'application. On décrit les activités, on donne quelques chiffres. On décrit ensuite le projet proprement dit, ainsi que la durée de vie qu'on envisage pour lui.

Grands choix techniques

On expose les différentes technologies informatiques employées dans l'entreprise pour les ressources (matériel, licence, expertise) ainsi que les projets d'évolution dans ce domaine.

Recueil des besoins fonctionnels

Sous forme textuelle toujours, on décrit les différentes tâches à réaliser dans le projet, avec le plus de détails possibles.

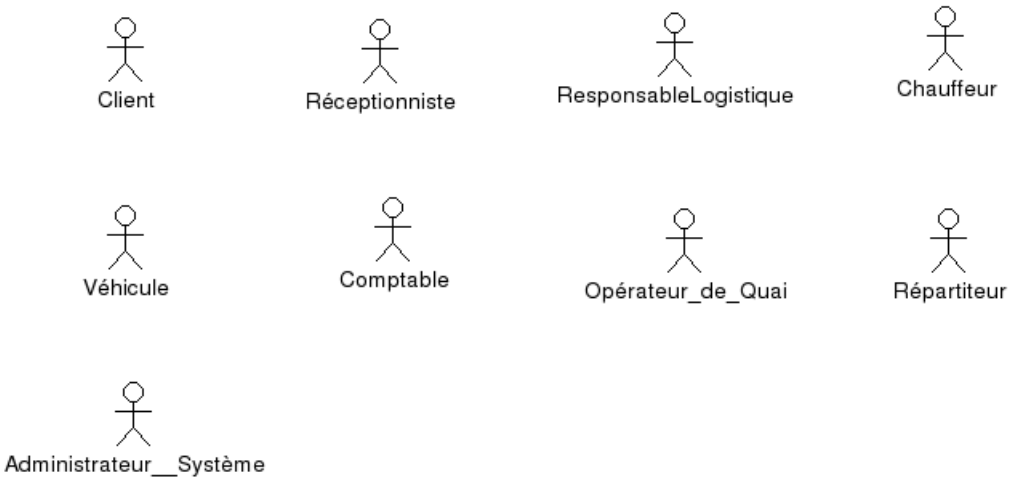
Recueil des besoins opérationnels

On fait intervenir des spécifications non fonctionnelles comme la vitesse, la sécurité et un examen du volume des données afin de pouvoir mieux définir par la suite les choix techniques.

2.2.2 Modélisation du contexte

On considère ici le système comme une boite noire à propos de laquelle on va
- identifier les entités externes qui interagissent avec le système (acteurs)
- répertorier les interactions entre acteurs et système
- représenter l'ensemble des interactions
  - modèle dynamique
  - modèle statique

Identification des acteurs



On va énumérer tous les acteurs, c'est-à-dire toutes les personnes amenées à interagir avec le système (le système ne doit pas figurer parmi eux, mais on pourra le retrouver dans les diagrammes de contexte). Quelques conseils :
- ne pas confondre personne physique et acteur : une même personne peut jouer des rôles différents à des moments différents. En général, plusieurs personnes peuvent incarner un même acteur (voir le diagramme de contexte statique qui sert précisément à quantifier les occurrences d'acteurs).
- si nécessaire, on peut stéréotyper les acteurs : acteurs internes (employés par l'entreprise) ou externes, acteurs non humains (une machine qui envoie des messages : exemple, un camion équipé d'un GPS qui téléphone sa position régulièrement)...

Identification des messages

L'énumération simple des messages entre les acteurs et le système constitue un point essentiel du travail. On pourra le représenter dans le diagramme de contexte dynamique. Une description détaillée de chacun de ces messages, qui constituent donc les interactions avec le système. Notons qu'il ne s'agit pas de cas d'utilisation (explicités dans une deuxième phase du travail), mais des messages émis lors de la réalisation des cas.

En principe, la spécification des messages entre acteurs constitue une perte de temps, puisqu'ils échappent à l'application.

Bouml n'offre pas de moyen confortable de gérer cette liste de message et surtout leur description. On pourra donc réaliser cette liste avec un éditeur de texte.

Réalisation des diagrammes de contexte

Diagramme de contexte dynamique Roques propose une utilisation (détournée) des diagrammes de communication : on y place les acteurs définis précédemment, ainsi que le système (*SIVEx* dans l'exemple). Une série de messages sont émis par les acteurs et par le système, qu'on décrit sommairement au moyen d'une expression courte. Le logiciel employé au cours oblige l'affichage de numéros séquentiels. On peut leur donner une signification temporelle (mais sans qu'elle soit vraiment stricte) ou l'ignorer sans plus (les messages sont alors numérotés pour simple référence, le cas dans l'exemple illustré ci-dessous).

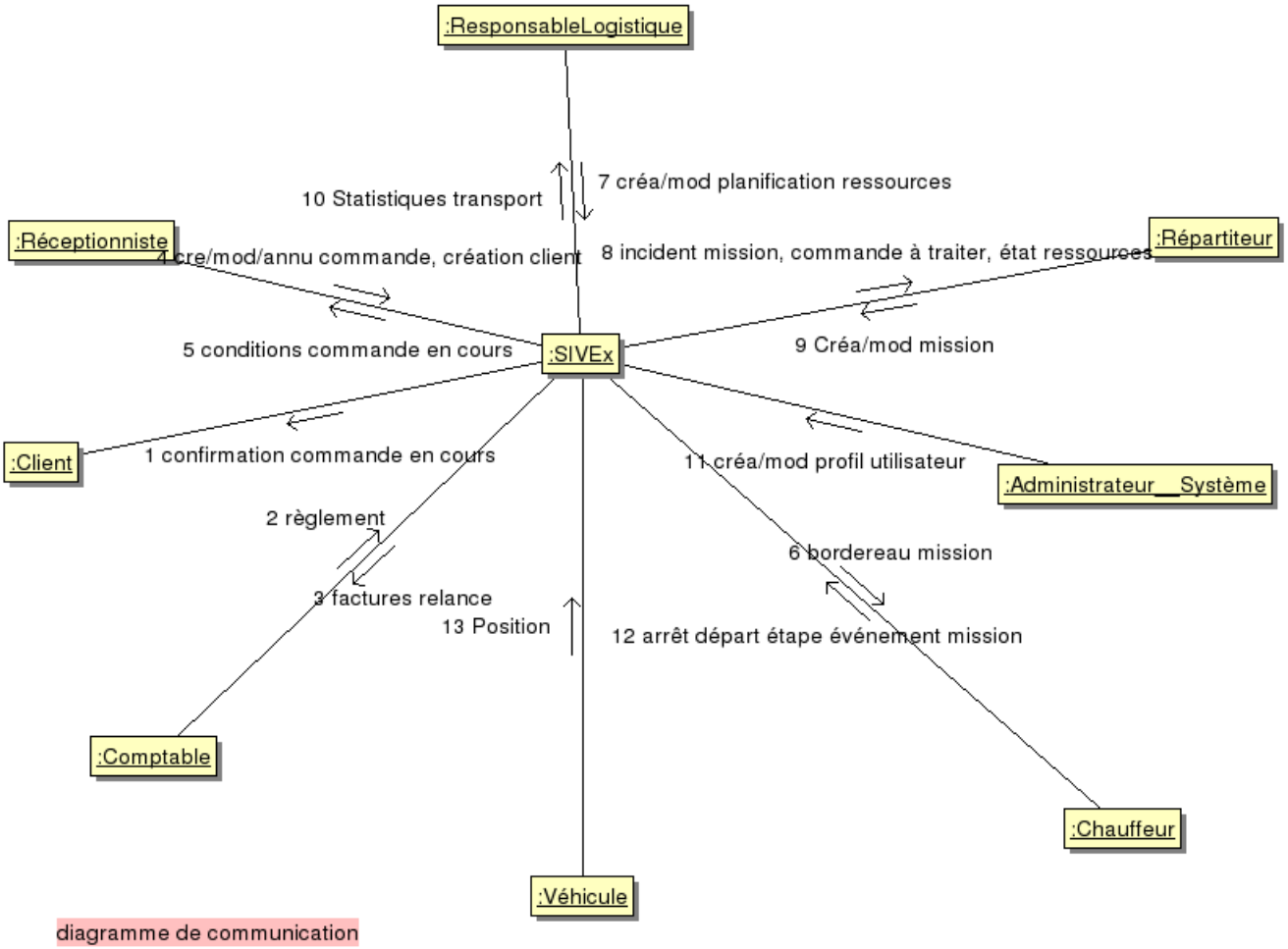
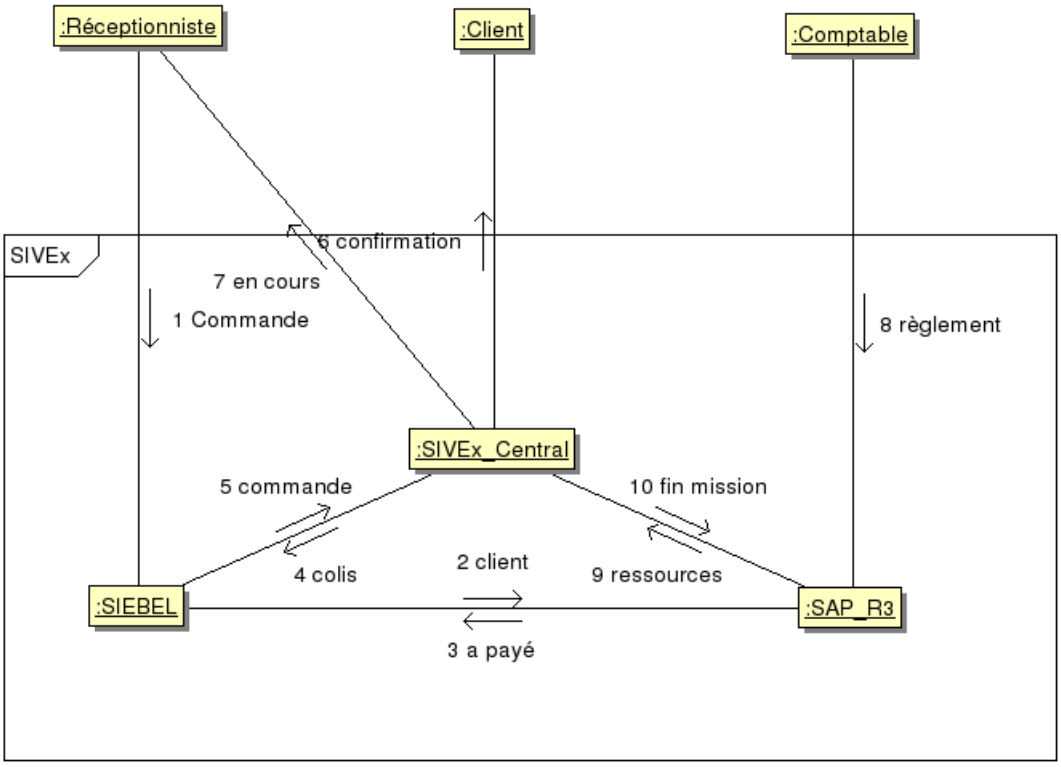


diagramme de communication

Dans des cas complexes, où interviennent plusieurs programmes, on peut faire apparaître une décomposition. Par exemple, outre le programme à écrire, ce diagramme intègre le logiciel Siebel (gestion des clients) et SAP (gestion comptable).



Modèle de contexte statique Il permet d'exprimer les multiplicités des différents acteurs. Ce diagramme perd de son utilité si les multiplicités sont toutes identiques (à 1 ou à plusieurs). Dans ce cas, il n'apporte aucune information supplémentaire par rapport au simple diagramme d'acteurs.

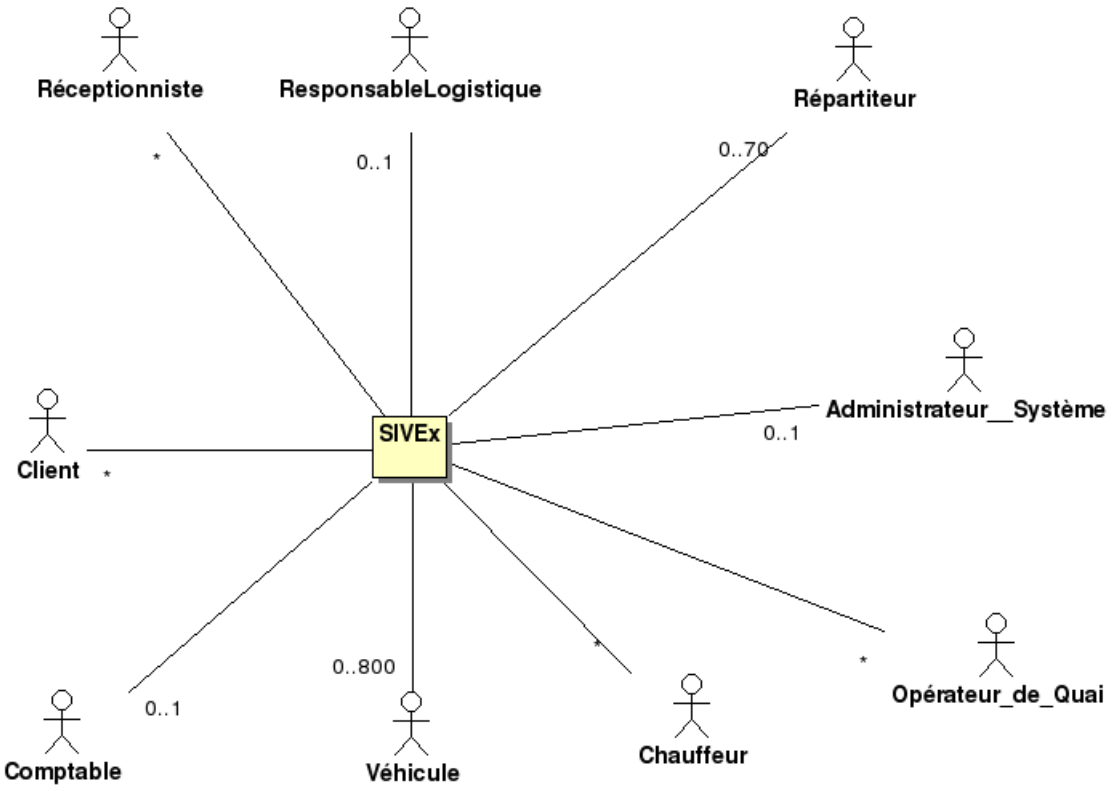


diagramme de classe