De la nécessité des cas d'utilisation

par Ivan Maffezzini

Il n'entendait rien à la médecine et appliquait ce maudit émétique à tous les maux. C'était sa panacée universelle. (G. Sand, citée dans le TLF)

L'observation scientifique est toujours une observation polémique ; elle confirme ou infirme une thèse antérieure, un schéma préalable, un plan d'observation […] *elle hiérarchise les apparences ; elle transcende l'immédiat ; elle reconstruit le réel après avoir reconstruit ses schémas. (*G. Bachelard, Le nouvel esprit scientifique*)*

1. Introduction

Même si tous ceux qui s'intéressent, de près ou de loin, au génie logiciel (GL) l'ont déjà souvent entendu, il vaut la peine de le répéter : ***un cas d'utilisation (CU) n'est pas un scénario*** ![1] Il vaut la peine de le répéter car, parfois, surtout dans des documents comme les Principes d'opération, on trouve un CU à côté d'un description fonctionnelle, là où, pour mieux clarifier les fonctions, il faudrait l'immédiateté d'un scénario bien choisi et non une duplication de la description fonctionnelle.

Étant donné l'importance des CU dans la majorité des approches qui emploient *Unified Modeling Language* (UML) et la diffusion de cette notation, nous allons extraire la définition de son manuel de référence [1] : « ***description d'un ensemble de séquences d'actions incluant des variantes qu'un système exécute*** ». Il s'agit de la définition du manuel de 1999, mais, bien qu'une définition fort différente de celle-ci soit contenue dans l'édition de 2005 du manuel, nos considérations seront fondées sur la définition la plus ancienne car, même dans un domaine qui « évolue » vite comme le GL, un lieu commun, pour être vraiment commun, a besoin de plus de quelques mois de vie (voir l'encadré 1 pour une comparaison des deux définitions).

Les trois citations suivantes, tirées de livres très populaires en GL, et qui aident les idées à se transformer en éléments communs, montrent que le syntagme « séquences d'actions » de la définition précédente est, en quelques sorte, un synonyme d'exigence fonctionnelle :

- Sommerville [2] : « ***les élicitations fondées sur les CU sont toujours plus employées pour l'élicitation des exigences*** [fonctionnelles] » ;
- Ghezzi [3] : « ***Les diagrammes CU décrivent le contexte global d'un système en subdivisant les fonctionnalités du système en transactions qui sont utiles pour les acteurs et qui montrent comment les acteurs interagissent avec eux.*** »
- Larman [4] « ***CU*** [sont] ***des exigences fonctionnelles qui indiquent ce que le système doit faire*** »

[1] Bien que tout le monde le sache, identifier les scénarios aux CU est une erreur si difficile à éradiquer que, dans la section sur l'élicitation des exigences du document qui présente l'état de l'art du GL, il est écrit que « *le type le plus commun de scénario est le CU* ». http »//www.swebok.org/

Une approche fondée sur les CU est donc une approche fondée sur les exigences fonctionnelles (le CU n'étant qu'une manière particulière de décrire les fonctions) dans le « contexte global d'un système ».

Et un scénario ? Toujours selon [1], un scénario est : « *Une séquence particulière d'actions qui décrit un comportement* ». En d'autres termes : une séquence particulière qui montre une instance d'un CU. En UML, le CU encapsule le comportement d'un élément du modèle — le sujet — dont il ne montre que les capacités de réagir aux messages envoyés par un acteur, où l'acteur « *est un ensemble cohérent de rôles que les utilisateurs des CU jouent lorsqu'ils interagissent avec les CU*[2] ».

Dans l'ingénierie, un outil et une méthode n'acquièrent une véritable utilité pour la « maîtrise » de la réalisation d'un produit que si on définit aussi *comment, quand, où* et par *qui* ils sont employés :

Comment ? Avec des spécification d'activités, des machines à état, des diagrammes d'interaction, des spécifications des antécédents et des conséquents, des textes en langue naturelle. Comme il est écrit en [1].

Où ? Dans les Principes d'opération et dans la Spécification des exigences système et du logiciel, si l'on emploie la terminologie de IEEE.

Quand ? La réponse à cette question n'est pas facile à moins de considérer une approche classique « bornée » (sans incréments et itérations) et une subdivision du temps ayant comme intervalles atomiques les temps de réalisation des artefacts. Dans ce cas lorsqu'on connaît le « où » (l'artefact) on connaît automatiquement le « quand ». Si on considère une granularité plus fine et des approches itératives et incrémentielles, le « quand » est l'un des éléments qui peut faire basculer le CU côté *honneur* ou côté *déshonneur*.

Qui : Pratiquement toutes les parties prenantes sont concernées (au moins en lecture), mais ce sont les utilisateurs, les experts du domaine et les analystes qui sont au centre de la rédaction.

2. Honneur

Même ceux, toujours plus nombreux, qui croient que les exigences non fonctionnelles sont souvent les exigences les plus contraignantes et les plus difficiles à satisfaire, ne peuvent s'opposer à l'affirmation que, sans une compréhension approfondie des fonctions, il est impossible de porter un projet à terme. Dans l'ingénierie en général, et dans le GL en particulier, tout le monde s'entend aussi sur le fait que la façon de décrire (ou de spécifier) les fonctions et les méthodes et les outils employés sont très importants.

Les divisions et les contrastes naissent lorsqu'il s'agit d'évaluer les méthodes et les outils employés pour décrire les fonctions.

---

[2] Définition, le moins que l'on puisse dire, bizarre.

Karl E. Wieger, dans la section « Avantages des CU » en [5], exprime, on ne peut plus clairement, les avantages d'employer les CU comme mécanisme de description des exigences fonctionnelles : « *La puissance de l'approche CU naît du fait qu'elle met au centre la tâche et l'utilisateur. Les utilisateurs auront des attentes plus claires de ce que le nouveau système leur permettra de faire que si vous prenez une approche mettant au centre les fonctions. […] Les CU aident les analystes et les développeurs à comprendre les règles d'affaires et le domaine d'application.* »

D'où découlent ces avantages ? Sans doute des faiblesses des descriptions fonctionnelles « classiques »[3] :

- il est parfois difficile de différencier ce que fait le système par rapport à ce que font les acteurs ;
- les échanges entre le système et les acteurs sont souvent mal formalisés ;
- la différence entre les rôles des utilisateurs n'est pas toujours claire ;
- les utilisateurs ont des difficultés à valider les fonctions, car les liens avec leurs actions ne sont pas toujours évidents.
- Les fonctions sont souvent trop abstraites.

Ces faiblesses de la solution « classique » sont pratiquement réglées avec une bonne application des CU — il serait plus correct de dire que les CU ont été introduits pour régler ces faiblesses. Voici une synthèse des avantages des CU en fonction des activités concernant les exigences.

- *Élicitation*. Le but de cette activité étant de comprendre ce dont les utilisateurs et les clients ont besoin, il est plus facilement atteint si on permet à l'utilisateur de « raconter l'histoire » de l'interaction qui a eu lieu ou qui a été envisagée avec le système. L'histoire, un ou des scénarios, est par la suite transformée en CU.
- *Catégorisation*. Dans cette activité, il s'agit de catégoriser les exigences selon des dimensions qui faciliteront la mise en œuvre : nécessité, priorité, stabilité, portée, etc. Le fait que l'on puisse dialoguer avec la bonne personne (celle qui est concernée par le CU et qui a donné le contenu à insérer dans le CU) rend la tâche aisée.
- *Modélisation conceptuelle*. Puisque chaque « acteur » parle de sa propre expérience concrète liée aux actions qu'il exécute, il est plus facile de mettre en évidence les concepts importants (pour l'utilisateur). On aura tendance à modéliser par « vue », ce qui rend la compréhension et la validation plus faciles.
- *Assignation des fonctions*. Puisque « l'acteur » qui est dans le scénario est celui-là même qui le raconte, il a donc les idées claires à propos des éléments avec

[3] Une description fonctionnelle « classique » étant, dans ce contexte, une description du comportement d'un système hiérarchisée et souvent obtenue à partir du flux des données.

lesquels il interagit. Et s'il ne sait pas très bien si c'est le matériel ou le logiciel, les sous-systèmes nº 1 ou le sous-système nº 2 qui doit lui répondre, il est facile de lui poser les bonnes questions.

- *Résolution de conflits*. Les conflits pour un rôle donné sont facilement réglables en questionnant en profondeur une personne et/ou en réalisant des entretiens avec d'autres intervenants qui peuvent jouer le même rôle. Les conflits entre rôles sont identifiés par les gestionnaires ou les responsables du système.
- *Spécification*. Toutes les considérations que nous venons de faire facilitent la rédaction des principes d'opération et des spécifications des exigences.
- *La validation des exigences*. Chaque acteur peut facilement valider ses CU et les gestionnaires ou les responsables systèmes peuvent valider l'ensemble, mais en sachant que chaque vue est, en principe, valide.

Mais ce n'est pas seulement dans le domaine de l'ingénierie des exigences que les CU sont des éléments utiles et efficaces. Ils facilitent aussi la conception système, les tests, la formation et la planification.

- *Conception système*. L'immédiateté du passage des CU aux diagrammes de séquences système[4] facilite l'écriture des contrats et donc la formalisation des réponses du système.
- *Tests*. Chaque action peut être considérée comme un élément de la conception des tests ayant un ensemble fonctionnellement cohérent de cas à tester. Comme l'écrit Perry William en [6] « *En introduisant les CU dans le cycle de développement, on fait plus facilement face aux cas de tests incorrects, incomplets et manquants*. […] *L'emploi d'une approche fondée sur les CU assure non seulement que l'on satisfasse les exigences mais aussi les attentes* ».
- *Formation*. La structuration par rôles favorise la production d'un matériel didactique orienté vers les caractéristiques des utilisateurs et une organisation de l'apprentissage et de l'enseignement par problèmes.
- *Planification*. Les CU, avec leur organisation en actions discrètes, sont l'un des intrants les plus sûrs et faciles à quantifier pour l'estimation des coûts des projets s'inscrivant dans une approche par points de fonction.

La force descriptive et prescriptive des CU dépend du fait qu'ils forcent les parties prenantes à définir les frontières (interfaces) entre les acteurs et le système et le contexte fonctionnel dans lequel le système opère dès les début du cycle de vie. Et c'est cette approche à boîte fermée renforcée par une mise en évidence des rôles et des séquences d'actions qui, en permettant de fractionner fonctionnellement le système, le rend plus facile à comprendre et à réaliser.

Par souci de clarté, nous avons commencé cette section en opposant les exigences fonctionnelles aux exigences non fonctionnelles, opposition quelque peu simpliste car

[4] Nous nous limitons à l'emploi de la notation UML.

une fonction « mal faite » est inutile (ou même dangereuse), tout comme l'est un système très performant ou facile à utiliser qui ne fait pas ce qu'il doit faire. Même s'il est vrai que les CU sont des descriptions fonctionnelles, leurs caractéristiques intrinsèques (immédiateté de la perception, discrétisation des actions, facilité de priorisation, etc), font qu'ils constituent l'un des facteurs essentiels pour livrer un produit de qualité. Ceci découle directement d'un cadre qualité comme celui de [7] : où le moteur de la qualité des artefacts est la qualité des processus. Qualité des processus qui, faut-il le souligner, est fortement influencée par la qualité des outils et des méthodes. Les CU sont une méthode/outils d'élicitation et d'analyse qui facilitent l'obtention de certains niveau de qualité sinon pour toutes les caractéristiques présentées en [8] au moins pour l'aptitude, l'exactitude, la capacité fonctionnelle et la facilité de test.

Trois derniers points en « l'honneur » des CU avant de terminer cette section :

1. Même si certains auteurs opposent une approche Stimulus-Réponse (S-R) aux CU il est clair que les CU aussi sont une description des réponses causées par des stimuli. On pourrait dire que les CU sont une approche (S-R) organisée par rôles.
2. Les CU ne se limitent pas à une approche par objets même si leurs éclosions ont eu lieu à peu près en même temps.
3. Il n'est pas vrai que, pour les interactions entre les humains et la machine, les CU aient besoin de la définition des interfaces (fenêtres, champs, clavier…). Les CU, comme les fonctions « classiques », peuvent — et doivent — être organisés en une hiérarchie qui commence avec les intentions des acteurs.

3 Déshonneur

Nous ne considérerons pas les critiques des tenants du développement agile, car nous avons déjà considéré leurs critiques dans la chronique du dernier numéro à propos de l'importance des modèles. Nous ne nous limiterons pas non plus à dire que les CU ne sont pas une panacée, car tout le monde sait que, dans une vie limitée par la technique, il n'y a pas de panacée, surtout pas de panacée universelle. Nous ne voulons pas non plus pinailler pour présenter des idiosyncrasies qui ne touchent que notre petit monde ou pour correspondre à des préoccupations académiques n'ayant aucun intérêt pour la pratique du GL.

Nous pourrions synthétiser les critiques qui sont adressées au CU (ou plus précisément aux tenants acritiques des CU) avec un lieu commun bien connu « les CU ont les défauts de leurs qualités ». Et, selon nous, c'est sa plus grande qualité (séparation nette entre ce que fait — ou ce qu'aimerait faire — l'acteur et ce que fait le système) qui est son défaut principal, quand on emploie les CU dans les toutes premières activités du GL.

Mais pratiquement toute la littérature insiste sur l'emploi des CU au début du CVL ! C'est bien là le problème.

Nous croyons que c'est UNE GRAVE ERREUR MÉTHODOLOGIQUE QUE DE COMMENCER PAR DÉCRIRE CE QUE LE SYSTÈME DOIT FAIRE. Ici, l'ambiguïté du terme « système » est particulièrement dangereuse et, dans la suite, nous emploierons donc le terme « machine ». Cela n'enlève aucune généralité à la critique, car le système, tel qu'il est considéré en [1] (surtout sans la version 2005), en [3] et en [4], et pratiquement dans toute la littérature, s'identifie à la machine à réaliser.

Puisque la machine que nous devons construire s'installe dans un DOMAINE AYANT DES LOIS INDÉPENDANTES DES EXIGENCES, la description de ce qu'elle fait doit être rédigée après l'élicitation et l'analyse des exigences et, surtout, après la modélisation du domaine : les exigences n'étant que les règles qui se transformeront en comportement de la machine pour contraindre certains phénomènes du domaine [8].

Confondre les lois du domaine avec les exigences ne peut que causer des problèmes lors de la construction et de l'exploitation de la machine. Les CU, en mettant les interactions au centre, mettent au premier plan les exigences fonctionnelles, et risquent ainsi de faire oublier les lois du domaine, ses contraintes[5] et les objectifs de qualité. À cela, on peut objecter que les « lois » du domaine ne sont généralement pas toutes écrites — surtout pas dans le marbre ! Et puis, même quand elles sont écrites, elles sont souvent d'accès difficile étant éparpillées dans plusieurs documents et il est donc impossible de les connaître sans interroger les utilisateurs. Objection accueillie ! Mais alors, pour connaître les lois selon l'angle des utilisateurs, ce ne sont pas les CU qui sont utiles, mais plutôt les scénarios. Dans un premier temps, il est préférable d'extraire de la concrétude des scénarios les concepts du domaine tels qu'ils sont envisagés par les utilisateurs plutôt que de les « desinstancier » dans les CU. Une fois que les concepts ont été extraits, les scénarios peuvent être « oubliés » et l'on peut travailler aux spécifications jusqu'à la conception système en mettant les concepts au centre. D'un point de vue méthodologique, les scénarios sont donc un instrument qui permet une percée dans les pratiques du domaine tel que perçu par les utilisateurs pour en extraire les concepts qui permettront de penser une nouvelle machine, éventuellement très différente de celle qui est connue ou envisagée par les utilisateurs. Les CU, en tant que généralisations hâtives des exigences fonctionnelles « pilotées par la machine connue », augmentent les risques de considérer comme des lois du domaine ce que n'est que concrétisation, dans la machine, d'exigences anciennes.

Vieille machine et nouvelle machine, dans la première phase du cycle de vie, s'opposent et doivent s'opposer à moins que cette dernière ne soit qu'une copie — améliorée, il faut l'espèrer ! — de la première. On suppose qu'il existe une vieille machine car, si elle n'existe pas les CU, ne décrivent que des interaction entre humains ; ce qui ouvre un chapitre trop complexe pour être abordé dans ce contexte. Elles s'opposent même quand il s'agit de changements dictés par des exigences de qualité car, dans le génie, deux machines qui exécutent les mêmes fonctions sont loin d'être la même machine. Décrire la « vieille » machine est très utile pour comprendre le domaine tel qu'il a été contraint par une mise en œuvre particulière, mais n'est

[5] Dans ce cadre-ci, il suffit de considérer les contraintes comme des lois moins figées que les lois du domaines mais plus rigides que les exigences.

pratiquement d'aucune utilité pour décrire les contraintes qu'imposera la nouvelle machine.

Ceux qui considèrent les CU comme un mécanisme pour éliciter les exigences (nouvelles) de la nouvelle machine se retrouvent dans une position inconfortable du point de vue logique : ils emploient les CU pour éliciter les exigences (nouvelles) et donc pour tracer les frontières de la nouvelle machine en décrivant la machine qui existe déjà comme si elle était, elle aussi, une loi du domaine et non un artefact particulier réalisé pour mettre en œuvre, de manière plus ou moins réussie, les exigences.

Comme l'écrit sans trop d'états d'âme M. Jackson : « *Le problème n'est pas à l'interface de la machine — il est profondément ancré dans le monde, bien loin de la machine.* » [8].

4 Au-delà

Nous avons l'impression que, pour aller au-delà des « observations polémiques » qui précèdent, il suffit d'aller dans le sens indiqué par le changement de définition dans le manuel de UML [1] que nous analysons brièvement dans l'encadré 1.

Il suffit de « faire glisser » le long de l'axe du temps le poids des CU comme nous le montrons dans la figure suivante, en nous inspirant des dessins qui accompagnent souvent la description du processus *Unified Process* prôné par beaucoup de tenants de la notation UML.

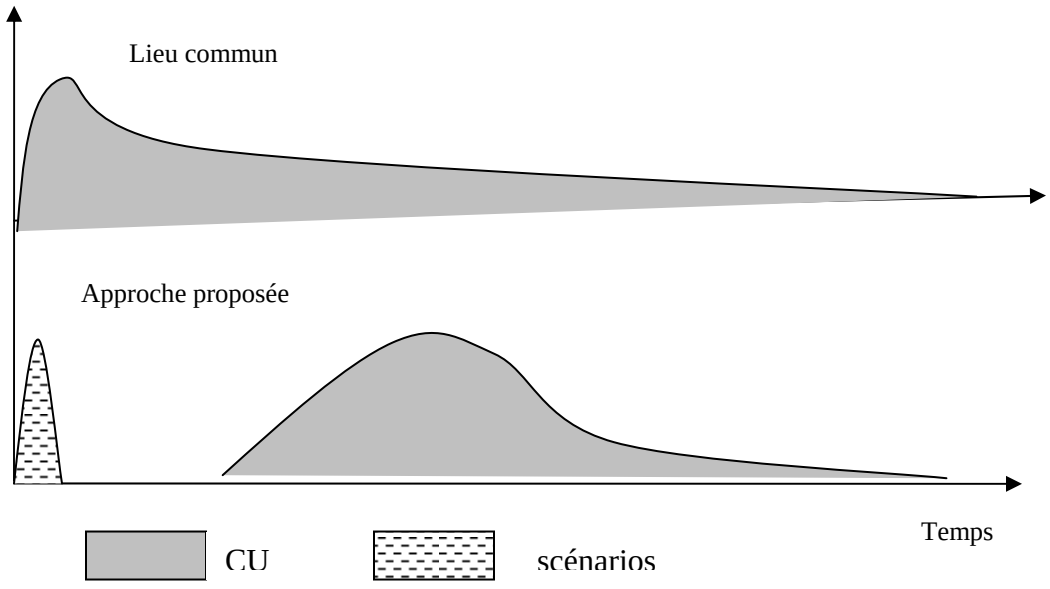

Figure 1 : CU et scénarios dans le temps

Les CU du « lieu commun » ont le maximum de ressources allouées au début du projet ; dans ceux de l'« approche proposée », le maximum glisse vers la conception (du système, des sous-systèmes et des classes). Dans l'approche proposée, nous avons

mis en évidence la courbe des scénarios que, dans la partie lieu commun, nous avons considérée comme étant complètement mélangée (au moins temporellement) aux CU.

Mais nous croyons que, pour donner aux CU leur vraie place, il ne suffit pas de les faire glisser vers la conception. Il faut aussi en relativiser leur importance qui, en ce moment-ci, est quelque peu exagérée[6]. Il faut considérer que les CU ne sont qu'une « manière de faire » parmi d'autres et que, dans le GL, il faudrait commencer à relativiser la « puissance » des « manière de faire » qui est influencée par un grand nombre de facteurs comme :

- Les caractéristiques du domaine ;
- Le niveau d'informatisation ;
- Les connaissances et les habitudes du personnel impliqué ;
- Le fait que les acteurs soient des machines ou des humains. À ce propos, nous croyons que le CU s'applique mieux aux projets dans lesquels les acteurs sont des machines[7].

Pour continuer sur cette lancée relativiste, nous ajouterons que l'efficacité d'un CU est très étroitement corrélée à la manière de le représenter : le fait de choisir une représentation textuelle simple, une représentation textuelle sur deux colonnes, une représentation graphique avec des diagrammes d'activités, d'employer des antécédents et des conséquences formels ou semi-formels, etc. a de grands impacts sur la facilité de compréhension, de vérification, de validation, de modification… La version du manuel UML de 2005 va dans ce sens en présentant cinq manières de les décrire parmi lesquelles le « texte en langue naturelle », qui est la manière consacrée de les décrire, apparaît en dernière position. Est-ce un hasard?

La lecture de livres et d'articles dithyrambiques sur les CU nous pousse à penser qu'il est au moins aussi naïf et peu scientifique de croire à une méthode universelle en GL que de croire à la pierre philosophale de nos prédécesseurs les alchimistes. La puissance d'un outil ou d'une méthode est, malheureusement, proportionnelle à l'étendue des domaines qu'ils visent à couvrir.

5 Références

[1] Booch G., Rumbaugh J., Jacobson I., *The Unified Modeling Language User Guide*, Second Edition, Addison-Wesley, 1999.

[2] Sommerville Ian, *Software Engineering*, Sixth Edition, Addison-Wesley, 2001.

[6] Dans notre expérience d'enseignement, nous sommes souvent confrontés à des étudiants qui, des activités d'analyse, n'ont pratiquement retenu que les CU (que, souvent, ils confondent avec les scénarios). Nous croyons qu'une analyse approfondie de ce phénomène permettrait non seulement d'améliorer les méthodes d'enseignement mais aussi de trouver certains fondements psychologiques du succès des CU qui pourraient contribuer à améliorer notre discipline. Voir en [9] une tentative de débroussailler.

[7] Auxquelles l'élicitation ne s'applique guère (au moins dans le sens psychologique du terme).

[3] Ghezzi Carlo, Jazayeri Mehdi, Mandrioli Dino, *Fundamentals of Software Engineering*, Second edition. Prentice Hall, 2003.

[4] Larman Craig, *Applying UML and Patterns*, Second edition. Prentice Hall, 2002.

[5] Wiegers, Karl, *Software Requirements. Second edition,* Microsoft Press, 2003.

[6] Perry William, *Effective Methods for Software Testing*, Addison-Wesley, 2000.

[7] ISO, ISO/IEC 9126-1 *Software engineering – Product quality : Quality model*, 2001.

[8] Jackson Michael, Problem Frames, Addison-Wesley, 2001.

[9] Ventimiglia Bernardo, Louis Martin, *A Case against de Use Cases in the Classroom ?*, http://www.trempet.uqam.ca/trempet/membres/Maffezzini/Articles/ArticlesGL/AgainstUsesCasesInTheClassroom.pdf, 1995.

Encadré No 1

Glissement de signification

La définition de CU dans les manuels de référence de UML a changé, selon nous de manière significative, de la version de 1999 à celle de 2005. Pour faciliter la comparaison, dans le tableau suivant, nous avons décomposé les deux définitions et catégorisé de manière qualitative leurs différences. La justification de la catégorie des différences — M(ajeures), (m)oyennes, (p)etites — est donnée dans la description qui suit le tableau.

No	Version 1999	Version 2005	Cat
1	La description	La spécification	m
2	d'un ensemble de séquences d'actions incluant des variantes	de séquences d'actions incluant des variantes et des séquences pour les erreurs,	p
3	qu'un système	qu'un système, un sous-système ou une classe	M
4	exécute	peut exécuter	m
5	et qui produit un résultat observable ayant une valeur pour un acteur	en interagissant avec d'autres objets dans le but de fournir un service utile	p

Tableau 1 : Comparaison des définitions de CU du manuel UML

1) Normalement, en GL, le mot « spécification » désigne des documents plus précis, plus détaillés, plus formels que ceux qui sont désignés par « description ». Le fait d'avoir introduit « spécification » va de pair avec les nouveaux éléments introduits en 3). À cause de la définition de spécification contenue dans le même manuel (*une*

description déclarative de ce que quelque chose fait ou est) nous avons placé ce changement dans les modifications m(oyennes)

2) Nous ne croyons pas que l'introduction d'« ensemble » change quelque chose à ce syntagme, par contre, l'introduction d'« erreur » nous semble être une autre indication qui va dans le même sens que « spécification » : on augmente la précision. Nous jugeons cette modification p(etite) parce qu'elle n'acquiert une certaine importance que si elle est considérée avec le passage de « description » à « spécification ».

3) Ici, nous n'avons aucun doute sur le fait qu'il s'agisse d'un changement M(ajeur), car l'ajout de nouveaux « sujets » (sous-système et classe) a d'énormes implications sur l'emploi des CU. Non seulement ces ajouts étirent son champ d'application mais ils l'étirent en lui faisant couvrir des activités de conception. NOTE. L'objection voulant que, dans la définition de 1999, le champ sémantique de « système » contienne aussi la signification des sous-système et de classe n'est pas recevable, car les lieux communs dont on parle, si bien colportés en [2], [3], [4] et [5], sont fondés sur la définition restreinte de « système »[8]. FIN DE LA NOTE.

4) Il nous semble que l'ajout d'une dimension de possibilité a une certaine importance, car elle augmente le poids du nouveau système par rapport au vieux.

5) Cette modification qui, de prime abord, semble importante ne l'est pas tellement, même si le passage de « acteur » à « objet » montre encore fois une tendance à généraliser les CU et à ne pas les « réserver » à l'analyse.

En résumé, il nous semble que tous les changements dans la définition vont dans le sens de donner une importance plus grande à l'emploi des CU dans la conception. (Ce n'est que lors de la conception que l'on a des sous-systèmes et des classes).

[8] Voilà un autre exemple de la confusion que le terme « système » sème un peu partout où il apparaît en GL.