Les règles et leurs transgressions aux différents niveaux d'abstraction des applications informatiques

Didier Sébastien

Les règles et leurs transgressions aux différents niveaux d'abstraction des applications informatiques

Didier SEBASTIEN, DOCTEUR
LIM-IREMIA, UNIVERSITÉ DE LA RÉUNION

CONTEXTE

L'Informatique est une science vaste qu'il est plus aisé d'étudier sous la forme de domaines fondamentaux. Parmi ces derniers, on peut distinguer le domaine applicatif qui se focalise sur l'aspect logiciel et qui servira de sujet d'étude dans le cadre de la présente communication sur la notion de « transgression ». En effet, le concept de transgression peut revêtir des aspects différents pour une application, en fonction de l'étape considérée de sa réalisation. Chacune de ces étapes est corrélée à un niveau d'abstraction plus ou moins élevé, allant d'une implication matérielle incontournable, jusqu'à une focalisation axée sur l'humain au travers d'une interface graphique mettant en œuvre des processus d'interaction homme-machine spécifiques.

Notre analyse de la transgression sur le domaine de l'application informatique sera décomposée selon les principales étapes suivies depuis la réalisation du logiciel jusqu'à son utilisation.

CRÉATION ET UTILISATION D'UNE APPLICATION INFORMATIQUE

Conception

La conception d'une application est la première étape de la création d'un logiciel. Elle est réalisée par le développeur et consiste à définir, à partir du cahier des charges, la logique de fonctionnement de l'application. En plus de la structuration du programme, relevant du génie logiciel (Strohmeier and Buchs, 1996), il s'agit de définir les algorithmes qui exécuteront les principales fonctionnalités de l'application. Ces algorithmes reposent sur un ensemble de règles logiques et mathématiques dont la transgression entraîne le non-aboutissement au résultat attendu.

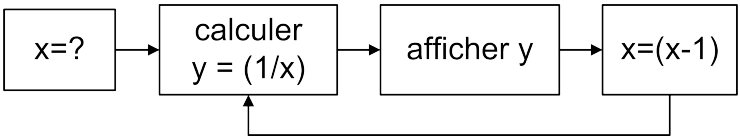

Figure 1. Exemple d'algorithme transgressif

La Figure 1 présente une schématisation de pseudo-code faisant état de deux types de transgressions différentes :

- mathématique, lorsque la valeur d'instanciation de la variable est comprise dans l'ensemble des entiers positifs ou nul ($x \in \mathbb{N}^{+}$), l'algorithme va rencontrer une erreur mathématique de division par zéro. Cette rupture de la logique calculatoire va provoquer une terminaison non-valide de l'algorithme.
- logique, le diagramme ne présente pas de condition de terminaison pour la boucle mise en place. Cette boucle infinie entraîne une non-terminaison du programme ($x \in \mathbb{N}^{-*}$).

Si à ce niveau conceptuel, l'humain peut concevoir un algorithme dont la terminaison est mauvaise (e.g. non respect du domaine de définition d'une fonction mathématique) ou absente (pas de condition pour la sortie de boucle), il n'en va pas de même pour l'ordinateur qui fera tourner cet algorithme. Nous pouvons donc considérer qu'à cette étape, une transgression des règles de conception est inenvisageable.

Codage et compilation

Le codage correspond à la traduction de la structure et de la logique de fonctionnement de l'application définie lors de la conception dans un langage de programmation particulier. Nous passons donc d'une description générique à une implémentation spécifique possédant son lexique et ses règles de syntaxe propres. À ce niveau d'abstraction, la logique calculatoire des données manipulées par le programme se retrouve imbriquée dans le code. Cette imbrication permet une meilleure gestion de son comportement et de sa possible transgression, à la condition qu'elle soit anticipée et traitée par le programmeur. La Figure 2 illustre au travers d'un pseudo-code le type de mécanisme qui peut être mis en place dans les langages évolués. Dans le cas présent, la division par zéro est détectée puis traitée séparément avant que le programme ne reprenne la trame principale des commandes étayant son exécution.

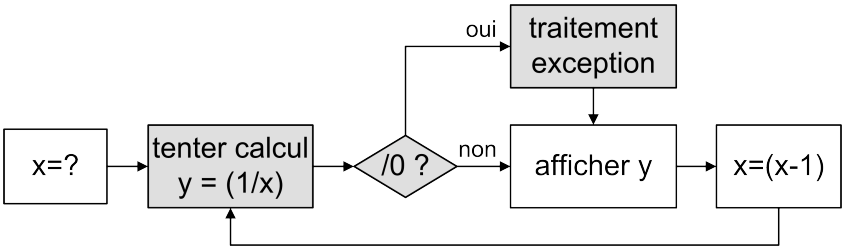

Figure 2. Traitement d'une exception sur un langage de haut niveau

La compilation (Gautier, 2006) est l'étape qui fait suite à l'écriture du code : elle permet, à partir de ce dernier, de créer un programme exécutable par l'ordinateur et donc fonctionnel pour l'utilisateur. Avant d'opérer cette transformation, le compilateur vérifie si le code produit par le programmeur respecte toutes les règles structurelles et syntaxiques rattachées au langage employé. Cette analyse ne transige sur aucune des règles et le blocage de ce processus peut intervenir pour une variation d'un seul caractère.

À ce niveau d'abstraction, on constate donc une forme de tolérance aux transgressions d'ordre mathématique sur les données manipulées, grâce à la mise en place de procédures de contournement, mais encore une fois une absence de tolérance aux transgressions des mécaniques permettant la manipulation de ces données.

Exécution

Si l'étape de compilation s'est déroulée convenablement, un programme appelé « exécutable » a été produit. En effet, à ce niveau d'avancement le programme peut maintenant être lancé par l'utilisateur final afin de profiter de ses fonctionnalités. L'exécution de l'application peut être observée sous deux aspects différents : du point de vue de la machine ou de celui de l'utilisateur (cf. partie suivante). Lorsque le programme tourne, son fonctionnement et l'ensemble des traitements qu'il opère sont assurés par le biais d'une unité de calcul : le processeur de l'ordinateur. Cette puce effectue à une fréquence élevée des opérations unitaires simples appelées « micro-instructions », qui mises bout à bout permettent les traitements avancés.

La nature et l'ordre des micro-instructions (Miller, Vandome and McBrewster, 2010) à exécuter sont transmis au processeur sous la forme d'un signal binaire : le code machine formé d'une suite de 0 et 1. Ainsi, à la manière du brin d'Acide Désoxyribonucléique (ADN) dont la succession de nucléotides décrit la suite d'acides aminés nécessaires à la formation d'une protéine, le code machine est composé d'un ensemble de micro-instructions décrivant la succession des opérations nécessaires au bon fonctionnement du programme. Mais ce parallélisme ne s'arrête pas là : de la même façon que l'ADN peut subir des

mutations par insertion, suppression ou remplacement d'une ou plusieurs de ses bases azotées, le code machine est susceptible de subir une modification dans sa séquence binaire. Ces dernières peuvent être causées par des paramètres physiques comme une température excessive du processeur, des problèmes d'isolation électro-statiques ou encore une pièce défectueuse dans l'architecture matérielle de l'ordinateur. La conséquence est qu'une ou plusieurs micro-instructions différentes de celles attendues seront réalisées, résultant souvent en une perte de l'intégrité mémoire des programmes en cours, voire du système d'exploitation hébergeant ces derniers. L'ordinateur interrompt alors son fonctionnement, effectue un vidage de la mémoire physique et redémarre.

Concernant le volet matériel de l'exécution d'une application, nous pouvons donc considérer qu'à ce niveau d'abstraction (bas), la transgression n'est pas admise car le système est peu en mesure de s'adapter à la perte de son intégrité mémoire.

Utilisation

Comme nous l'avons dit, l'exécution d'une application peut être observée sous deux aspects : du point de vue de la machine, mais également du point de vue de l'humain qui fait emploi du programme. Dans le cadre de cette étude nous nous intéresserons à deux niveaux d'utilisation différents du programme. Nous commencerons par évoquer le cas de l'interface graphique environnant l'application avant de nous intéresser à un exemple de contenu évolué de très haut niveau : un environnement virtuel immersif.

Interface graphique

Le niveau de satisfaction sur une expérience d'utilisation d'une application dépend d'un ensemble de critères comme la correspondance entre la nécessité et l'atteinte de l'objectif recherché via son emploi, la richesse des fonctionnalités proposées ou encore l'aisance à la manipulation. Dans ce sens, la notion d'ergonomie traduit la justesse, l'optimisation et la logique de l'Interaction Homme-Machine (IHM) (Sharp, Rogers and Preece, 2007). L'une des composantes importantes de l'ergonomie est liée à la notion d'« *expectancy* » (littéralement « attente » ou « espoir ») ; qui caractérise le fait que l'icône/bouton activé par l'utilisateur déclenche bien l'action qu'il souhaite effectuer. Pour une application nouvelle, il s'agit donc d'une hypothèse formulée *a priori* sur la base de l'interprétation du pictogramme recouvrant un bouton.

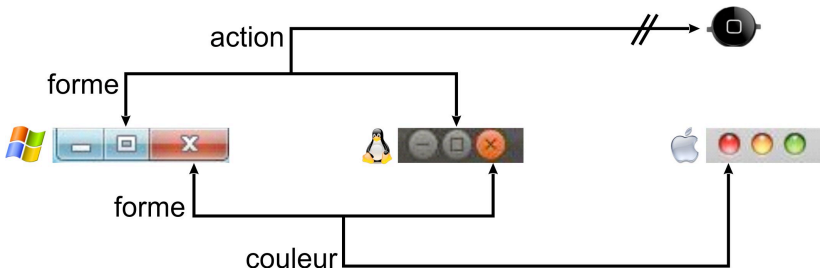

Figure 3. Comparaison d'éléments d'IHM de plusieurs systèmes

Prenons ainsi l'exemple des boutons de contrôle des fenêtres sur trois des principaux systèmes d'exploitation. La Figure 3 illustre le fait que la forme en croix (×) est couramment utilisée pour signifier la fermeture définitive d'une fenêtre. La signification de la forme est par ailleurs augmentée d'une couleur rouge dont le consensus est plus large et correspond à une symbolique d'alerte courante en Occident. Dans un contexte usuel, bien que ce soit possible, on peut difficilement concevoir qu'une transgression de ce code puisse paraître naturelle ou comprise par l'utilisateur. Ainsi, ce dernier perdrait ses repères si le concepteur du logiciel décidait de reprogrammer les boutons des fenêtres afin que la croix corresponde au redémarrage de l'ordinateur.

Pour autant, des exemples de la plasticité humaine sur les pratiques en IHM existent et s'expliquent par la contextualisation des usages. Cette dernière permet de redéfinir les codes en vigueur avec une modification de la perception du signe. C'est le cas de l'icône rectangulaire qui permet usuellement de maximiser la taille d'une fenêtre sur les systèmes d'exploitation, mais qui correspond à un retour au bureau sur les terminaux portables de marque Apple™. Dans ce dernier cas, le contexte d'utilisation modifié est rapidement intégré par l'utilisateur qui n'expérimente plus cette modification de la relation signe/fonction comme une transgression.

Dans ce sens, nous pouvons remarquer la capacité humaine à redéfinir ses règles entre signifiant et signifié pour ne pas expérimenter des interactions nouvellement usuelles comme des transgressions.

Environnement immersif

Dans cette dernière partie, nous nous intéressons au niveau d'abstraction le plus élevé d'une application : l'utilisation directe du contenu du programme et l'expérience qu'il engendre. Naturellement, ce contenu dépend de la nature de l'application et il en existe des plus ou moins complexes. Pour cette étude nous prendrons pour exemple un cas avancé de programme plaçant l'utilisateur dans un environnement tridimensionnel immersif et temps réel. Ce type de monde virtuel, également appelé méta-univers (ou plus simplement « métavers ») permet

de plonger l'utilisateur dans un monde conçu sur mesure qu'il peut explorer avec une vue à la première personne très proche de la vision humaine (Smart, Cascio and Paffendorf, 2007).

Il est alors intéressant de comparer les propriétés particulières des mondes virtuels qui dénotent autant de possibles transgressions par rapport au monde réel. En effet, pour un métavers :

- l'espace exploitable est illimité. Si la surface disponible pour une construction réelle est généralement contrainte, les environnements virtuels ne prennent en considération ce paramètre qu'au moment de leur affichage, uniquement lorsque les ressources matérielles en termes d'architecture de calcul sont réduites. Avec la démocratisation des supports de calculs puissants et de moteurs d'affichage de représentations 3 D optimisés, la contrainte informatique matérielle est de plus en plus obsolète. La conception d'un métavers, et par là la créativité du designer, n'est donc pas impactée par une limite spatiale.
- les normes de sécurité ou encore les contraintes physiques architecturales n'ont pas besoin d'être respectées. Par exemple, il est possible de faire un escalier sans balustrade ou superposer des étages sans multiplier les piliers porteurs.
- l'ajout d'un élément est moins onéreux que dans le monde réel. Cela correspond à la modélisation d'un objet tridimensionnel et de ses fonctionnalités par un infographiste.
- l'accès est généralement bien plus aisé que pour un lieu du monde réel. Ainsi, les méta-univers sont à même de proposer des reproductions d'environnements difficiles d'accès ou dangereux. Les technologies actuelles permettent maintenant de reconstituer une zone physique à partir d'un modèle numérique de terrain décrivant un relief et d'une orthophotographie fournissant la texture qui s'y superpose. Un tel procédé rend accessible avec un niveau de précision notable, même aux personnes à mobilité réduite, l'expérience d'une exploration telle que celle de la surface de la lune.
- l'adaptabilité de l'environnement est poussée. En effet, l'espace est entièrement reconfigurable à partir du déplacement des différents objets qui le composent. Il est par exemple possible d'augmenter l'espace entre deux bâtiments sans les détruire pour en intercaler un troisième.
- les éléments qui le composent sont réutilisables. Une fois un modèle tridimensionnel créé, il peut être dupliqué en fonction des besoins et pour une utilisation dans un contexte différent.

La possibilité de s'affranchir des limitations précédentes permet de concevoir et expérimenter des espaces complètement dédiés à un objectif. L'imagination du concepteur n'est pas bridée par des contraintes physiques annexes, seules comptent la performance et l'adéquation de l'expérience transmise à l'effet recherché. Cet effet peut être de natures diverses : artistique, ludique, mais également pédagogique (Sebastien, Conruyt and Rakotobe, 2012) et thérapeutique. L'utilisation professionnelle des mondes virtuels est en pleine expansion, avec la multiplication des « *serious-games* » (Susi, Johannesson and Backlund, 2007), qui reprennent les mécaniques ludiques des jeux mais dans un contexte sérieux et pour favoriser la transmission d'un savoir-faire ou diminuer un facteur de stress. Ainsi, la formation des pilotes d'avions passe immanquablement par des simulateurs capables de reproduire à la demande des configurations critiques : l'accident est alors pédagogique et les pertes matérielles inexistantes. De la même façon, l'intérêt du monde médical pour les environnements immersifs n'est pas en reste. Ils font l'objet d'expérimentations probantes parmi lesquelles on peut citer le traitement et l'atténuation des symptômes relatifs aux phobies : en exposant virtuellement les patients dans des configurations plus ou moins menaçantes, il est possible d'améliorer leur confiance face à ce qu'ils perçoivent comme un danger. De la même façon, les personnes ayant subi d'importantes brûlures se trouveront apaisées dans un métavers figurant un monde glacé.

La capacité du virtuel à violer les règles matérielles et physiques du monde réel constitue dans un sens une transgression de haut niveau d'abstraction qui peut se révéler bénéfique pour son utilisateur. Il s'agit alors d'une exploration imaginaire dont résulte une expérience humaine concrète, orientée pour être positive et focalisée sur un besoin. La transgression revêt alors un caractère souhaitable et utile.

CONCLUSION ET PERSPECTIVES

La notion de transgression pour les applications informatiques se définit par rapport aux règles caractéristiques des étapes de leur développement et utilisation, eux-mêmes liés aux niveaux d'abstraction qui les caractérisent. Plus ce niveau est bas et proche de la machine et moins la transgression est acceptable. L'exception est au mieux tolérée si cette dernière a été anticipée et qu'un mécanisme de prise en charge est prévu. En effet, la logique des langages informatiques évolués est à même de protéger les données qu'elle manipule, mais elle se révèle incapable d'appréhender sa propre substance afin de sauvegarder son intégrité : il lui manque la conscience d'elle-même pour accepter la transgression.

À l'inverse, lorsque le niveau d'abstraction est élevé et que l'attention est portée sur l'humain, son interaction avec la machine, alors la transgression peut être acceptée puis redéfinie pour devenir la règle dans un contexte clairement

identifié. Enfin, lorsque l'on se concentre sur l'expérience apportée par des contenus du plus haut niveau, la transgression se révèle pédagogique voire cathartique, dans le sens où elle permet de se constituer un vécu réel orienté pour être positif sur la base d'un environnement virtuel.

Les perspectives découlent naturellement des limitations mises en exergue dans le cadre de la présente communication. S'il n'est évidemment pas possible à l'heure actuelle de transgresser les règles logiques et mathématiques, il serait intéressant d'améliorer la réponse des couches basses à une exception inattendue (Dong, Pérez-Cortés and Collet, 2002). Indépendamment des travaux permettant la vérification des programmes (Payet and Mesnard, 2006), la multiplication des niveaux d'abstraction et la méthode du « *sandboxing* » permettant l'isolement des programmes exécutés, offrent de nouvelles garanties sur la fiabilité logicielle. Il est également envisageable de séparer matériellement les ressources mémoires de la plate-forme d'exécution des logiciels qui s'exécutent, afin de limiter les conséquences d'une transgression et permettre un fonctionnement en mode dégradé. Il faut que la machine, comme l'humain, soit en mesure de conscientiser puis borner précisément sa transgression pour enfin l'accepter.

BIBLIOGRAPHIE

DONG, P.-Q., PÉREZ-CORTÉS, E., and COLLET, C., « La tolérance aux fautes adaptable pour les systèmes à composants : application à un gestionnaire de données », actes des Journées des Bases de Données Avancées, 2002.

GAUTIER, M., *Compilation des langages de programmation : Ce que fait un compilateur, comment le réaliser*, Paris : Ellipses Marketing, 2006.

MILLER, F. P., VANDOME, A. F., and MCBREWSTER, J., « Fat Binary: Computer program, Instruction set, Machine code, Operating system, Source code, Installation (computer programs), Compiler, Virtual machine », 2010.

PAYET, E., and MESNARD, F., « Non-termination inference of logic programs », *ACM Transactions on Programming Languages and Systems*, 28, Issue 2:256-289, March 2006.

SEBASTIEN, D., CONRUYT, N., and RAKOTOBE, M. H., « The neuron-based architecture for making smart campuses as pedagogical immersive environments », Proceedings of the 5th Conference on eLearning Excellence in the Middle East, eLex 2012, 30th January-2nd February, Dubai, 2012.

SHARP, H., ROGERS, Y., and PREECE, J., *Interaction Design: Beyond Human-Computer Interaction*, England : John Wiley & Sons Ltd, 2007.

SMART, J., CASCIO, J., and PAFFENDORF, J., « Metaverse Roadmap, Pathways to the 3 D Web. A Cross-Industry Public Foresight Project », online resource of the Metaverse Roadmap project, 2007.

STROHMEIER, A., and BUCHS, D., *Génie logiciel : principes, méthodes et techniques*, Lausanne : Presses Polytechniques et Universitaires Romandes, 1996.

SUSI, T., JOHANNESSON, M., and BACKLUND, P., « Serious Games - An overview », Technical report, HS-IKI-TR-07-001, University of Skövde, 2007.