Zeste de savoir

Que ferait l'utilisateur ? — 97 choses à savoir quand on est un programmeur

22 mars 2019

Table des matières

1. Introduction

Introduction

Il existait, il y a de cela deux-trois ans, un site dont le nom était « 97 Things Every Programmer Should Know » mais qui aujourd'hui semble avoir disparu (bien qu'on puisse trouver le Gitbook). Ce site était composé de **conseils divers et variés sur la programmation**, écrits par différents programmeurs et librement diffusé sous licence Creative Commons Attribution Non Commercial Share Alike 3.0 . Je me permets donc de partager ici avec la communauté de Zeste de Savoir des traductions de ces différents articles. N'hésitez pas à commenter.

L'article traduit aujourd'hui est Se demander « Que ferait l'utilisateur ? » par Giles Colborne.

2. Se demander « Que ferait l'utilisateur ? »

Nous avons tendance à présumer que **les autres pensent comme nous**. Mais ce n'est pas le cas. On appelle ça l'effet de faux consensus . Quand des gens pensent ou agissent différemment de nous, inconsciemment nous sommes tentés de les classer comme déficients, d'une certaine façon.

Cet effet explique pourquoi les programmeurs ont tant de mal à se mettre à la place des utilisateurs. Ces derniers ne pensent pas comme les premiers. Déjà, ils passent beaucoup moins de temps derrière un ordinateur. Ils ne connaissant pas, et bien souvent ne s'intéressent pas, au fonctionnement d'un PC. Cela signifie qu'ils ne peuvent pas se reposer sur toutes les techniques de résolution de problèmes si familière aux développeurs. Ils ne reconnaissent pas les patterns et les repères avec lesquels les programmeurs ont l'habitude de travailler sur, autour et à travers une interface.

Le meilleur moyen de comprendre comment pense un utilisateur est encore d'en regarder un. Demande à un utilisateur de faire une tâche qui utilise un morceau de logiciel similaire à celui que tu développes. Sois bien sûr que la tâche à réaliser est bien réelle : « Additionne une colonne de nombres » c'est OK ; « Calcule tes dépenses du mois passé » c'est mieux. Évite les tâches

trop spécifiques : « Peux-tu sélectionner ces cellules du tableau et entrer une formule SUM en-dessous ? » — il y a un indice dans la question. Fais parler l'utilisateur pendant qu'il fait ce que tu lui a demandé. Mais ne l'interromps pas et ne l'aide pas. Plutôt, continue de te demander « Pourquoi est-ce qu'il fait ça ? » et « Pourquoi est-ce qu'elle ne fait pas ça ? »

FIGURE 2. – CommitStrip l'a encore une fois bien illustré, l'application que l'utilisateur ne comprend pas.

La première chose qu'on remarque, c'est que les utilisateurs ont tendance à **regrouper les**

actions similaires. Ils tentent de remplir les tâches dans le même ordre et font les erreurs aux mêmes endroits. Il faut établir le design en se basant sur ce comportement central. C'est différent des réunions de design où l'on entend bien souvent « Et si jamais l'utilisateur veut … ? » Cela mène à des fonctionnalités élaborées et de la confusion par rapport à ce qu'ils veulent. **Observer les utilisateurs élimine cette confusion**.

Tu vas voir les utilisateurs se retrouvés coincés. Quand tu es coincé, tu regardes autours. Quand ça arrive aux utilisateurs, leur attention se réduit. Il leur devient plus difficile de trouver la solution ailleurs sur l'écran. C'est une des raisons pour lesquelles un texte d'aide n'est qu'une mauvaise solution à un mauvais design d'interface. Si tu es obligé de mettre des instructions ou un texte d'aide, sois sûr de **le placer juste à côté de l'endroit problématique**. C'est parce que l'attention de l'utilisateur est réduite que des *tool tips* sont plus utiles que des menus d'aide.

Les utilisateurs ont tendance à se débrouiller. Ils trouveront un moyen de faire ce qu'ils veulent et s'en contenteront, peu importe à quel point ça peut être alambiqué. Il est donc préférable de fournir un moyen évident de faire les choses plutôt que deux ou trois raccourcis. Tu verra qu'il y a aussi un fossé entre ce que les utilisateurs disent vouloir et ce qu'ils font. Le meilleurs moyen de comprendre ce qu'ils veulent est donc de **les observer**. Regarder comment procède les utilisateurs pendant une heure est plus instructif que de passer une journée à deviner ce qu'ils veulent.

3. Mon mot à moi

Designer une interface est un travail délicat et c'est pour ça qu'il existe des gens spécialisés dans l'UX. Nous, programmeurs, n'avons pas peur des interfaces complexes, des lignes de commandes, des menus très longs, des raccourcis, etc. Il suffit de regarder n'importe quel IDE, il est rempli de menus et sous-menus. Mais l'utilisateur lambda n'est pas comme ça. Il convient donc de bien viser son public pour ne faire ni trop complexe, ni trop simpliste.

Un reproche avait, par exemple, était fait à l'interface de rédaction de ZdS, que certains jugent peu claire. Des personnes ont ainsi avoué ne pas avoir eu le courage d'écrire des contenus pour cette raison.

? Votre parole

Et vous, qu'en pensez-vous ?