

ce

Interface PM
IFT2905
Fuille de notes

Franz Girardin

18 février 2024

Table des matières

2	Chapitre 1 Why Design fails & Design of Everyday Things	
3	Chapitre 2 Design Centré sur \mathcal{U}	
3	Chapitre 3 Évaluation heuristique	
3.1	Évaluation heuristique	3
4	Chapitre 4 Design centré sur les tâches	
4.1	Workflow du design centré sur les tâches	4
4	Chapitre 5 Les exigences du design	
5	Chapitre 6 Modèles conceptuels (MC)	

Why Design fails & Design of Everyday Things

Mythe de l'erreur humaine \mathcal{H} Les échecs d'un système PM sont souvent dus au design. Pour \downarrow erreur \mathcal{H} :

- Design qui tient compte des **limitations** et de la **fiabilité** des \mathcal{H} .

Principes de design

- Utilisabilité
- Expérience de l'Utilisateur (UX)
- Psychopathologie** : frustrations courantes
- Permettent de **critiquer**, **analyser** et **converger** interfaces.

Causes d'échecs

- Fonctionnalité
 - \mathcal{U} ne connaît pas fonctions de l'Obj.
 - L'objet ne fait pas ce que \mathcal{U} désire.
- Visibilité
 - \mathcal{U} voit pas certaines infos l'Obj.
 - \mathcal{U} ne sait pas quelle séquence de contrôle est nécessaire pour atteindre son but.
 - E.g. Lumières enfoncée pour passage piétons
- Feedback
 - Comment \mathcal{U} sait si les opérations ont réussi?
 - Comment \mathcal{U} sait s'il y a une erreur en cours de route?

Buts du UX | MAUSSEE : Mémorabilité, Apprentissage, Utilité, Sécurité, Satisfaction, Efficience, Efficacité.

Définition de l'utilisabilité Degré selon lequel un produit peut être utilisé par des \mathcal{U} identifiés, pour atteindre des **buts définis** par l'**efficacité**, l'**efficience** et la **satisfaction**.

- Efficacité** : atteindre le but ☉
- Efficience** : effort et ou temps minimal ⌚
- Satisfaction** : évaluation subjective par \mathcal{U}

Note :

L'utilisabilité implique aussi la sécurité, l'apprentissage et la mémorabilité.

Où les designers se trompent

- Ne comprennent pas \mathcal{U} et leurs limitations
- Ne prévoient pas différents **contextes d'utilisation**
- Absence de **modèle détaillé** du fonct.
- Absence de **feedback** par l'objet.

Pourquoi le design est-il difficile Les interactions sont complexes et difficile à définir. Par ailleurs, les tâches sont complexes et *implicites*. Il faut distribuer *raisonnablement* les tâches à la machine et à l' \mathcal{U} pour éviter que l'un ou l'autre ne soit pas confronté à une complexité excessive.

Principe de découvrabilité L' \mathcal{U} doit savoir immédiatement à quoi l'objet sert, comment l'utiliser et quelles sont les opérations possibles.

- Affordance** ce que l'O permet de faire. Un *signifiant* est un élément qui permet de rendre l'affordance visible.
- Signifiants** indiquent que l'affordance \exists et ne doivent pas être **contradictoire**.
- Anti-affordance** permettent de masquer visibilité d'un aff. et contribue à la gestion d'erreur. Il s'agit d'une affordance qui est *délibérément supprimée*
- Correspondance** Permet d'établir la relation entre une action et la réaction de l'interface.
 - Soit une série de lumières alignées et des interrupteurs un à côté de l'autre, quel interrupteur allume quelle lumière.
- Contraintes** sont des restrictions physiques ou logiques de l'objet ou l'interface qui contraignent l' \mathcal{U} à utiliser l'objet d'une certaine façon. P. ex., orientation d'une *clé USB*.
- Feedback** permet d'indiquer à l' \mathcal{U} l'effet de son action ou d'une interaction.
- Modèle conceptuel** est une explication très simplifiée du fonctionnement d'un élément.
 - Modèle fonctionnel** : on sait *quoi* faire sans savoir *pourquoi* ça marche
 - Modèle structurel** : on connaît les composants et leurs interactions
 - Exemple : *Chinese mystery pot*

Les deux fossés d'interaction La conception doit permettre de résoudre 2 ensembles de questions que l' \mathcal{U} se pose lorsqu'il interagit :

- Comment ça marche et *qu'est-ce que je peux faire avec l'objet ?*
- Qu'est-ce qui s'est passé et *est-ce que c'est ce que je voulais faire ?*

Les sept étapes d'une actions 1. Définir l'*objectif*, 2. Former l'*intention* (ce qu'on s'apprête à faire pour atteindre l'objectif) 3. Spécifier la séquence d'actions, 4. Exécuter l'action 5. Percvoir l'état du système 6. Interpréter l'état du système 7. Évaluer l'état du syst. *par rapport aux intentions*.

Analyse de la cause originelle Permet de déterminer si un *objectif* est réelle la fin souhaitée ou est simplement un sous-objectif d'un but à atteindre.

Sept questions garantissant l'atteinte de l'O Un *bon design* implique qu'à tout moment, l' \mathcal{U} parvient à répondre aux **sept questions** suivantes.

- Que puis-je faire, 2. Quelles sont les alternatives 3. Comment puis-je le faire, 4. Le fais-je bien ? 5. Qu'est-ce que ça veut dire 6. Que s'est-il passé ?

Mythe de l'erreur humaine

- Importance du design dans la réduction des erreurs humaines

Principes de design

- Utilisabilité
- Expérience de l'Utilisateur (UX)
- Psychopathologie

Causes d'échecs

- Fonctionnalité
 - Connaissance des fonctions par l'utilisateur
 - Adéquation des fonctions aux besoins de l'utilisateur
- Visibilité
 - Visibilité des informations
 - Clarté sur les séquences de contrôle nécessaires
- Feedback
 - Indication de la réussite des opérations
 - Signalement des erreurs

Buts du UX

- Mémorabilité, Apprentissage, Utilité, Sécurité, Satisfaction, Efficience, Efficacité

Définition de l'utilisabilité

- Efficacité
- Efficience
- Satisfaction

Où les designers se trompent

- Compréhension des utilisateurs et de leurs limitations
- Prévision des différents contextes d'utilisation
- Modélisation détaillée du fonctionnement
- Feedback de l'objet

Pourquoi le design est-il difficile

- Complexité des interactions et distribution des tâches

Principe de découvrabilité

- Affordance
- Signifiants
- Anti-affordance
- Correspondance

- Contraintes
- Feedback
- Modèle conceptuel (fonctionnel et structurel)
- Les deux fossés d'interaction
 - Compréhension du fonctionnement et validation des actions
- Les sept étapes d'une action
 - Définition de l'objectif à l'évaluation des résultats
- Analyse de la cause originelle
 - Détermination des objectifs réels et sous-objectifs
- Sept questions garantissant l'atteinte de l'objectif
 - Questions clés pour un bon design

Section

2

Design Centré sur \mathcal{U}

Risque du modèle en cascade Il permet de s'assurer que les implémentations sont *conformes aux exigences*, mais ne garantit pas qu'elles sont optimales pour \mathcal{U} . Par ailleurs, les problèmes sont parfois identifiés plus tard et revenir en arrière dans la modèle cascade peut être **couteux**

- Problèmes identifiés tard.
- Manque d'input et *feedback* de l'U
- Problèmes → modif. exigences et conception.

Design itératif Étale le projet sur plusieurs petites itérations de *conception*, *prototypage* et *évaluation*.

Modèle en spirale Il utilise le principe de design itératif et réduit graduellement les risques à travers les *itérations*. Seules les itérations matures sont présentée à l' \mathcal{U} .

Design centré sur l' \mathcal{U} Marque un changement de paradigme où l'opinion de l' \mathcal{U} a précédence sur la *technologie* ou l'intuition du designer. On conçoit en fonction de ce que les \mathcal{U} *doivent, peuvent ou veulent* faire.

- **Prédesign** pour comprendre le problème
- **Premier design** pour explorer l'espace de design
- **Mi-design** Développe l'approche choisie
- **Design avancé** lorsqu'on intègre et déploie L'évaluation par les \mathcal{U} se fait de façon *continue*; ils accompagnent les développement à chaque itération :

Design Il faut 1. analyser les utilisateurs, les tâches *qu'ils cherchent à accomplir*; il faut com-

prendre le problème et s'assurer que l'idée de solution est importante ou au moins *nécessaire* pour les \mathcal{U} . Il faut estimer le niveau d'expertise des \mathcal{U} . Il faut aussi 2. suivre les principes de conceptions liés à l'*utilisabilité* Finalement, il faut 3. assurer la prévention et gestion des erreurs.

Implémentations brouillons Elle peut être sur papier ou de style *Wizard of Oz* C'est *rapide, simple* et suffisamment *abstrait* pour se concentrer sur l'essentiel.

Évaluation Permet de relier la *progression de la conception* aux *besoins identifiés* et aux contextes de l'utilisateur. L'évaluation peut être effectué *tôt* ou *tard*, selon le besoin.

Identifier les parties intéressées Cela dépend de plusieurs questions :

Parties intéressées

- qui l'utilisera?
- qui décidera de l'utiliser?
- qui va payer pour cela? 2. qui doit le faire (concevoir / construire)?
- qui doit en tirer profit?
- qui rendra votre vie misérable

Section

3

Évaluation heuristique

Types d'erreurs

➤ **Bévue** et **Lapsus** sont causés par *manque réflexion*

- **Bévue** → échec d'exécution.
- **Lapsus** → faille de mémoire lors de l'exec.
- **Méprise** est un échec de mémoire et ou planif. P. ex., planifier procédures selon objectif, mais procédures utilisées ne mènent pas à l'objectif (méprise sur le choix de procédure).

Type d'erreurs

- Erreur d'exécution
 - Lapsus
 - Perte d'intention
 - Omission due à l'interruption
 - Omission due à la satisfaction
 - Bévue
 - De Capture
 - De Description
 - De Mode
- Méprises

➤ **Erreur de capture** Survient lorsqu'un *comportement fréquent* capture notre attention et nos action. P. ex. vouloir taper *facial recognition* sur Google et finir par naviguer sur facebook.

➤ **Erreur de description** Survient lorsqu'une *description* n'est pas suffisamment précise. P. ex. confondre les logos **Reply** et **Replay** All.

➤ **Erreur de mode** Survient lorsqu'on utilise un *contrôle* dans un contexte différent de celui qu'on souhaite. P. ex. utiliser **j** ou **k** dans d'autres application que Vim.

Stratégies pour prévenir les erreurs

- **Bévue** Éviter actions aux démarches similaires
- **Lapsus** Fonctions de *forçage*
- Obliger retrait carte ATM pour prendre \$
- **Erreur de mode** Éliminer mode ou ↑ *visibilité* modes.

Stratégies générales

- Désactiver *commandes illégales*
- Rendre informations *nécessaires* visibles
- Préférer les listes déroulantes aux zones de texte
- Utiliser les *dialogue de confirmation*
- Vous êtes sur le point de **rm -rf**!

Contrôle par l'utilisateur et liberté L'interface doit encourager l'exploration en rendant les choses visibles et en assurant que les conséquences des erreurs ne soient pas sévères.

Exploration ⇒ ↑ Découvrabilité

➤ S'il y a **automatisation**, les changements doivent aussi pouvoir être effectués manuellement. L'automatisation ne doit pas **override** l'autonomie de \mathcal{U} . Les saisies (**input**) doivent être. Un \mathcal{U} doit pouvoir créer, modifier, lire, mettre à jour et supprimer toute donnée. **modifiables**

Le droit d'annuler implique que L' \mathcal{U} doit pouvoir décider; les opérations doivent être annuables, et toutes boîtes de dialogue doit avoir un bouton d'annulation, pour ne pas **piéger**

3.1 Évaluation heuristique

Avantages Peut être fait rapidement, sans équipement, sans \mathcal{U} et pour peu de \$.



Étapes d'une évaluation heuristique

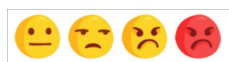
- **Mise en place** préparation du matériel (scénarios, prototype, liste d'heuristiques)
- **Évaluation** pas exploration de l'interface, application des heuristiques génération de liste de problèmes
- **Agrégation** retour et discussion entre experts.

Note :

Il est important d'avoir **plusieurs évaluateurs** parce chacun d'eux peuvent identifier *différents problèmes*

Évaluation Il faut au moins **2 passages** par évaluation ; le premier sert à découvrir le flux de la plateforme et le second permet de *cibler* des éléments spécifiques.

➤ **Liste de problèmes** est produite par l'évaluateur ; il y mentionne *l'heuristique qui est violée* et attribue un *cote* de gravité.



- **Impact du problème**
- **Fréquence du problème**

Marqueurs de bon design Les *U* n'attendent pas trop longtemps et on offre du *feedback* (visibilité) ; on utilise des mots, symboles et conventions qui *match* avec le monde réel, p.ex *trashbin* ; on *ne force pas* l'utilisateur sur un chemin fixe ; on permet la révision, correction, modification, suppression ; on *rend les erreurs impossibles* et valide les entrées ; on s'assure que les tâches courantes se font de façon efficiente grâce aux accélérateurs de clavier et souris, abréviations, double-clic, raccourcis de menu et touches de fonction, etc. ; on aide l'*U* à reconnaître, diagnostiquer et récupérer les erreurs ; on offre de la documentation.

Section 4

Design centré sur les tâches

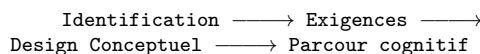
Persona Représentation symbolique de l'*U* décrivant ses caractéristiques et les *tâches* qu'il doit accomplir.

Parties intéressées Stakeholders Réfère à toute personne ayant une *raison* de se soucier à l'interface.

- Souvent nombreux
 - ▶ Les *U*
 - ▶ Les dev.
 - ▶ Les constructeurs
 - ▶ Les sponsors

- Intérêts parfois en conflits
 - ▶ Ces efficience, utilité
 - ▶ Facilité de dev.
 - ▶ Facilité de construction
 - ▶ Coût de financement

4.1 Workflow du design centré sur les tâches



Identification Identifier l'*U* ; analyser des *exemples de tâche*

Exigences Décider quelles tâches le design soutiendra.

Note :

Il faut décrire les tâches à l'aide d'**exemple de tâches imaginées** pour parvenir à déterminer les exigences. Les tâches peuvent être décrites par leur fréquence et leur importance.

En pratique, pour identifier les tâches, on peut contacter les potentiels *U* et discuter avec eux du fonctionnement du système pour avoir leur feedback. Il faut demander à l'*U* de réviser les exemples de tâches :

- Omissions
- Corrections
- Clarifications
- Suggestions

Clarification de la distinction tâche-activité Dans le *design centré sur les tâches*, le mot *tâche* a une signification plus proche du mot *activité*, puisque le design comprends prend en compte une grande quantité de tâche et leur contexte d'exécution.

Bon exemples de tâches Ils sont décrivent *quoi* sans extrapoler *comment* ; il sont *spécifiques* ; ils sont *exhaustif* et décrivent le travail de l'*U* dans son ensemble ainsi que la transmission globale de l'information. Dans l'ensemble, ils reflètent les intérêt des vrais *U*.

Design conceptuel Décrire la base d'interaction et de représentation (modèle mental). On utilise un *scénario* :

tâche + design = scénario

Le scénario montre comment une tâche est gérée par le design.

Tâche : Elle choisit une recette dans la liste

Scénario : Elle fait *défiler la liste* d'images et *tape sur...*

Parcours cognitif Parcourir les tâches *pas à pas* en utilisant le design et *évaluer*.

Avantages de design centré sur les tâches

- Permet de baser le design sur *U* et *tâches réelles*
- Permet d'identifier les exigences lors de la *phase pré-design* (avant même design conceptuel).

Les exigences du design

Types d'exigences

- **Fonctionnelle** : ce que le produit doit faire
 - ▶ *U* choisit une *recette* sur le site.
- **De données** : caractéristiques des données requises
 - ▶ *U* choisit une recette sur le site en regardant une *liste*
- **D'environnement** : Contexte physique et non physique nécessaires au fonctionnement.
 - ▶ P. ex. une *mère célibataire* qui *travaille à domicile*
- **Propres à l'*U***
 - ▶ P. ex. une *mère célibataire* a des *enfants* qu'elle garde à la *maison* lorsqu'elle navigue sur le site.
- **De but d'utilisabilité ou d'expérience *U***
 - ▶ Elle navigue sur le site et *trouve un recette pour préparer événement spécial*

Comment identifier les exigences

En théorie, il faut trouver :

- ▶ Les besoins humains
- ▶ Les *U* et parties intéressées
- ▶ Les niveaux de supports nécessaires (à l'aide des *métriques*)

Les métriques Elles aident à identifier le niveau de support nécessaire (l'attention et les ressources à accorder) sur un aspect du produit.

Qualitatives

- ▶ Historique des impressions et frustrations
- ▶ État actuel des impressions et frustrations
- ▶ Documentation des incidents critiques
- ▶ Effet viral
- ▶ Possibilité de personnaliser, Récupération d'erreur, Rétention des compétences acquises, Facilité de réorganiser des activités, etc.

Quantitatives

- Vitesse
- Taux d'erreur
- Simplicité d'apprentissage
- Facilité d'utilisation

Astuce pour les formulaires d'inscription

- Cacher les affordances causant des erreurs
 - P. ex. bouton pour réinitialiser les entrées du formulaire
- Utiliser un label plutôt qu'un placeholder
 - Le texte du placeholder disparaît ;
 - Il a un mauvais contraste
 - Ne permet pas à l'U de valider son entrée
 - Nécessite mémoire à court terme de l'U
 - Les l'U peuvent confondre la texte du placeholder avec une info entrée
 - etc.

- Expliquer toutes les exigences d'entrées
 - P. ex. mdp doit contenir des caractères spéciaux

- Supprimer les champs optionnel tant que possible
 - Ne pas faire un sondage lors de sign-up

- Indiquer clairement quel champ sont optionnel
 - P. ex. avec astérisque

- Faire correspondre les champs au type et à la taille de l'entrée
 - P. ex. éviter les listes déroulantes lorsqu'il y a peu d'options Préférer les radio button

Section 6

Modèles conceptuels (MC)

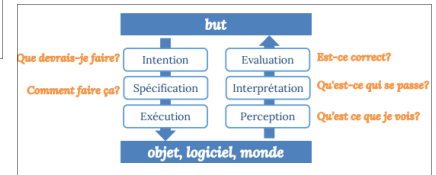
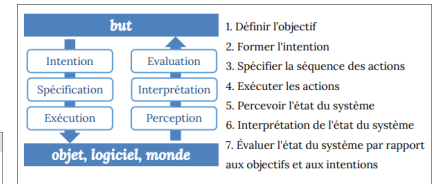
Rappel Il s'agit de l'idée que l'U se fait de la façon dont fonctionne un objet ; sont modèle peut être *fonctionnel* (c.-à-d. l'U sait **quoi** faire) ou *structurel* (c.-à-d. l'U sait comment **parts** interagissent) MC peut être incomplet et quand même être précis/utile.

Rappel : Modèle mentaux (MM)
Modèles que les gens se font du fonctionnement des objets qu'ils utilisent et qui leur permet de prédire le fonctionnement des futurs objets avec lesquels ils interagiront.

- En voyant une **paire de ciseaux**, nos MM nous permettent de déduire : affordances, signifiants, contraintes, correspondances, feedback, modèle conceptuel.

Exécution et débogage d'un MM
Permet d'établir la relation de cause à effet entre **action** et sa **conséquence** ; permet de faire des **prédictions** et de générer des **explications** quant au fonctionnement.

Sept étapes de l'action On parcourt les 7 étapes pour déboguer le MM

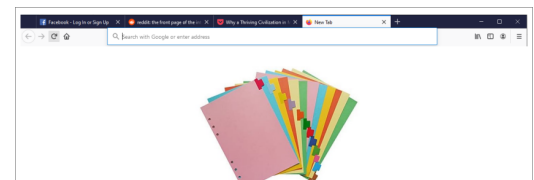


Caractéristiques d'un bon MC

- Basée sur les tâches
- Intuitif et cohérent
- Facile à apprendre
- Utilise une terminologie claire

Éléments définissant un MC

- **Métaphores** : elles permettent de communiquer (intuitivement) à l'U avec peu d'explication de quoi il s'agit
 - P. ex. bureau = **desktop** ⇒ un des espaces de travail principaux.
- **Concept, actions, rôles de l'U et attributs.**
- **Correspondance entre les concepts et l'UX**
- **Les types d'interactions**
- **Les types d'interface**



Exemple de métaphore

- **Onglet = page**
- **contenu = texte**
- **Actions**
 - Lire
 - **Onglet = page Copier**
 - **Onglet = page Ouvrir un nouvel onglet = nouvelle page**
- **Comment**
 - **mouse click**
 - **keyboard**

Important de connaître type d'erreur différentes (e.g. lapsus vs. bévues) ? Différencier définir objectif vs. former intention dans les 7 étapes d'une actions.