IHM

Interface Homme-Machine
Quand le doigt rencontre l'écran

Par William Cloudless

Table des matières

IHM

L'interaction humain-machine (IHM) est l'étude des interactions entre les utilisateurs et les ordinateurs. Elle se concentre sur la conception, l'évaluation et la mise en œuvre de systèmes informatiques interactifs destinés à être utilisés par des humains.

Histoire

Avant JavaFX, il existait l'AWT (Abstract Window Toolkit). C'est la première bibliothèque graphique Java, dépendante de la plateforme. Par la suite, Swing est une bibliothèque graphique qui est une amélioration de l'AWT. Celle-ci est indépendante de la plateforme. Enfin, JavaFX. Il est plus performant et plus moderne.

Elément dit « nœud »

Widget

Un widget est un composant graphique interactif, tel qu'un bouton, une zone de texte ou une liste déroulante. Les widgets sont utilisés pour construire l'interface utilisateur d'une application.

Liste des widgets :

Widgets de saisie de texte:

- **TextField**: Permet à l'utilisateur de saisir une seule ligne de texte.
- **TextArea**: Permet à l'utilisateur de saisir plusieurs lignes de texte.
- **PasswordField**: Masque les caractères saisis par l'utilisateur, utilisé pour les mots de passe.

Widgets de sélection:

- **Button**: Un bouton cliquable.
- **RadioButton**: Permet à l'utilisateur de choisir une seule option parmi un groupe.
- **CheckBox**: Permet à l'utilisateur de sélectionner ou désélectionner une option.
- **ChoiceBox** et **ComboBox**: Permettent à l'utilisateur de choisir une option dans une liste déroulante. ComboBox permet également la saisie de texte.
- **ListView**: Affiche une liste d'éléments et permet la sélection d'un ou plusieurs éléments.
- **TreeView**: Affiche une hiérarchie d'éléments et permet la sélection.
- **Slider**: Permet à l'utilisateur de sélectionner une valeur numérique en faisant glisser un curseur le long d'une barre.
- **ScrollBar**: Permet de faire défiler le contenu d'une zone lorsque celui-ci dépasse les limites de la zone visible.
- **DatePicker**: Permet à l'utilisateur de sélectionner une date.
- **ColorPicker**: Permet à l'utilisateur de choisir une couleur.

Autres widgets:

- **Label**: Affiche un texte non modifiable.
- **Tooltip**: Affiche une info-bulle lorsque l'utilisateur survole un nœud avec la souris.
- **Hyperlink**: Un lien hypertexte cliquable.
- **ImageView**: Affiche une image.
- **WebView**: Affiche du contenu web (pages HTML).
- **ProgressBar**: Indique la progression d'une tâche.

Gestionnaires de placement

Les gestionnaires de placement sont des composants qui permettent de positionner les widgets dans une fenêtre. Ils permettent de définir la taille et la position des widgets, ainsi que leur comportement lorsque la fenêtre est redimensionnée.

Liste des gestionnaires :

- **BorderPane**: Divise l'espace en cinq régions : haut, bas, gauche, droite et centre. Idéal pour les mises en page classiques où vous avez une barre de menu en haut, une barre d'état en bas, etc.
- **FlowPane**: Les nœuds enfants sont placés dans le sens du flux (horizontal ou vertical) les uns à côté des autres ou les uns en dessous des autres, en créant automatiquement de nouvelles lignes ou colonnes si nécessaire.
- **HBox**: Place les nœuds enfants horizontalement, les uns à côté des autres.
- **VBox**: Place les nœuds enfants verticalement, les uns en dessous des autres.
- **TilePane**: Organise les nœuds enfants dans une grille où toutes les cellules ont la même taille.
- **GridPane**: Organise les nœuds enfants dans une grille flexible où les nœuds peuvent occuper plusieurs cellules.
- **StackPane**: Empile les nœuds enfants les uns sur les autres, ce qui est utile pour créer des superpositions ou des effets visuels.
- **AnchorPane**: Permet de "fixer" les bords des nœuds enfants à une distance spécifique des bords du conteneur, ce qui est pratique pour créer des interfaces réactives.

Programmation événementielle

La programmation événementielle est un paradigme de programmation dans lequel le déroulement d'un programme est contrôlé par la survenue d'événements. Les événements peuvent être générés par l'utilisateur (par exemple, un clic de souris), par le système (par exemple, un changement d'état d'un périphérique) ou par le programme lui-même. Lorsqu'un événement se produit, le programme exécute une fonction appelée gestionnaire d'événements qui est associée à cet événement.

Fonctions de rappel

Les fonctions de rappel (callbacks) sont des fonctions qui sont enregistrées dans un widget et qui sont appelées lorsque l'une des opérations du widget est activée. Par exemple, lorsqu'un bouton est cliqué, la fonction de rappel associée au bouton est appelée.

Événements

Les événements peuvent être liés aux périphériques (par exemple, un clic de souris, une frappe au clavier) ou aux applications (par exemple, la création ou la destruction d'une fenêtre). Lorsqu'un événement se produit, il est placé dans une file d'attente. La boucle de gestion des événements prend les événements dans la file d'attente et les traite.

Les événements avec JavaFX

Dans JavaFX, un événement est un objet qui hérite de la classe `javafx.event.Event`. Tout événement a trois propriétés :

- Une source : le nœud qui a généré l'événement
- Une cible : le nœud qui doit traiter l'événement
- un type : le type d'événement (par exemple, `ActionEvent, MouseEvent, KeyEvent`)

La réponse à un événement peut se faire par un gestionnaire d'événements ou par un filtre d'événements.

Classes d'événements

Il existe différentes classes d'événements pour une même action physique. Par exemple, un clic de souris sur un bouton génère un `ActionEvent`, tandis qu'un clic de souris sur un rectangle génère un `MouseEvent`.

Route d'événements

Lorsqu'un événement est généré, les actions suivantes sont réalisées :

1. Sélection de la cible de l'événement.
2. Construction de la route de l'événement. La route est une liste de nœuds qui seront traversés par l'événement.
3. Traversée de la route. L'événement est propagé à travers les nœuds de la route, en commençant par la racine de la scène et en descendant jusqu'à la cible de l'événement.

Abonnement et notification

Pour qu'un nœud puisse traiter un événement, il doit s'abonner à cet événement auprès de l'abonneur. L'abonneur est le nœud qui gère les événements. Lorsqu'un événement se produit, l'abonneur notifie tous les abonnés en leur transmettant l'événement. Chaque abonné peut alors traiter l'événement.

Widgets et événements

Interface EventHandler

L'interface **`EventHandler<T extends Event>`** est une interface fonctionnelle qui permet de définir des gestionnaires d'événements. Elle ne contient qu'une seule méthode abstraite, **`handle(T event)`**, qui est appelée lorsque l'événement se produit.

Gestionnaires d'événements

Il existe plusieurs façons de définir des gestionnaires d'événements en JavaFX :

- Classe externe : une classe qui implémente l'interface **`EventHandler`**.
- Classe interne : une classe définie à l'intérieur d'une autre classe.
- Classe interne anonyme : une classe interne sans nom.
- Expression lambda : un raccourci de syntaxe pour les classes internes anonymes.
- Implémentation directe de l'interface **`EventHandler`**: la classe qui contient le gestionnaire d'événements implémente directement l'interface **`EventHandler`**.

Utilisation de setOnXXX

JavaFX fournit des méthodes `setOnXXX` pour enregistrer un gestionnaire d'événements sur un nœud. Par exemple, la méthode `setOnAction` permet d'enregistrer un gestionnaire d'événements pour l'événement `ActionEvent`.

MVC (Modèle-Vue-Contrôleur)

MVC (Modèle-Vue-Contrôleur) est un motif de conception logicielle qui permet de séparer les données d'une application, leur présentation et les interactions de l'utilisateur. Le modèle contient les données de l'application, la vue affiche les données à l'utilisateur et le contrôleur gère les interactions de l'utilisateur.

ListView

Le widget ListView permet d'afficher une liste d'éléments à l'utilisateur. Il utilise deux modèles :

1. `ObservableList` : Contient les données de la liste (les éléments à afficher).
2. `SelectionModel` : Gère la sélection des éléments dans la liste.

ListView utilise un système de listeners (observateurs) pour être notifié des changements dans le modèle de données (`ObservableList`) et mettre à jour l'affichage en conséquence.

Canvas

Canvas est une classe JavaFX qui permet de dessiner directement sur une zone de l'interface utilisateur. On peut dessiner des formes géométriques basiques, du texte, des images et manipuler des pixels. Tous les dessins sont effectués dans un contexte graphique (`GraphicsContext`) associé au Canvas. Le système de coordonnées utilisé place l'origine (0,0) en haut à gauche du Canvas.

Feuilles de style (CSS)

JavaFX permet de personnaliser l'apparence des éléments de l'interface utilisateur à l'aide de feuilles de style CSS. Cela fonctionne de la même manière que les CSS pour les pages web HTML. On peut utiliser des fichiers CSS externes ou définir les styles directement dans le code Java.

Conception et prototypage

Conception centrée utilisateur (CCU)

La CCU est une approche de conception qui place les besoins et les attentes des utilisateurs au centre du processus de développement. Elle vise à créer des interfaces plus utilisables, c'est-à-dire plus efficaces, efficientes et satisfaisantes pour les utilisateurs. La norme ISO 9241-210 définit l'utilisabilité et les facteurs qui y contribuent.

Étapes clés de la conception d'IHM

1. Comprendre et spécifier le contexte d'utilisation:
    - Identifier les utilisateurs cibles et créer des personas (profils d'utilisateurs fictifs).
    - Comprendre les tâches que les utilisateurs effectueront avec le système.
    - Tenir compte de l'environnement technique et organisationnel.
2. Spécifier les exigences utilisateur et l'objectif d'utilisabilité:
    - Définir des objectifs d'utilisabilité précis et mesurables (temps d'exécution des tâches, taux d'erreur, satisfaction, etc.).
    - Prioriser les exigences en fonction de leur importance.
3. Concevoir des solutions:
    - Générer des idées de conception (brainstorming).
    - Créer des prototypes (maquettes) pour visualiser et tester les solutions.
4. Évaluer les solutions:
    - Tester les prototypes avec des utilisateurs représentatifs.
    - Recueillir les commentaires des utilisateurs et les analyser.
    - Itérer sur la conception en fonction des résultats de l'évaluation.

Prototypage

Le prototypage consiste à créer des représentations limitées d'un produit ou système afin de tester et d'évaluer des idées de conception avant de les implémenter complètement. Les prototypes peuvent prendre différentes formes, allant de simples maquettes papier à des logiciels interactifs.

Types de prototypes

- **Prototypes papier (basse fidélité)**: Maquettes dessinées à la main ou créées avec des outils simples. Ils sont rapides et peu coûteux à produire, mais offrent une interactivité limitée.
- **Prototypes numériques (moyenne fidélité)**: Créés avec des outils de conception graphique ou de prototypage. Ils offrent une meilleure représentation visuelle et une certaine interactivité, mais peuvent prendre plus de temps à produire. (Figma)
- **Prototypes fonctionnels (haute fidélité)**: Logiciels partiellement fonctionnels qui simulent l'expérience utilisateur finale. Ils sont les plus coûteux à produire, mais offrent le niveau de réalisme le plus élevé. (SceneBuilder)

Type de schéma

- **Séquences d'interaction**: Représentent les différentes étapes d'une interaction utilisateur avec le système, souvent sous forme de diagrammes de flux.
- **Schémas d'interface**: Représentent la disposition visuelle des éléments de l'interface utilisateur (boutons, menus, champs de texte, etc.).

Heuristiques et recommandations

Il y a différentes heuristiques et recommandations pour la conception d'interfaces utilisateur (IU) efficaces et conviviales.

Les 10 heuristiques de Nielsen

Jakob Nielsen, expert en utilisabilité, a proposé dix heuristiques qui sont largement utilisées pour évaluer et améliorer la conception d'interfaces utilisateur.

- Proposer des dialogues simples, naturels et minimum.
- Parler le langage de l'utilisateur.
- Minimiser la charge de mémoire de l'utilisateur.
- Être cohérent, respecter les standards.
- Refléter le monde réel.
- Fournir un retour à l'utilisateur.
- Permettre une personnalisation pour les utilisateurs expérimentés.
- Prévenir les erreurs.
- Aider l'utilisateur à reconnaître, diagnostiquer et réparer les erreurs.
- Rendre accessible aide et documentation.

Les 8 critères ergonomiques de Bastien et Scapin

Ces critères fournissent un cadre d'évaluation plus détaillé pour l'ergonomie des interfaces utilisateur.

- **Guidage**: L'interface doit guider l'utilisateur à travers des messages clairs, des regroupements visuels, un retour d'information immédiat et une bonne lisibilité.
- **Charge de travail**: L'interface doit minimiser la charge cognitive de l'utilisateur en étant concise, en réduisant les actions nécessaires et en présentant l'information de manière claire.
- **Contrôle explicite**: L'utilisateur doit avoir le contrôle sur ses actions et sur le système, avec des actions explicites et la possibilité d'interrompre ou d'annuler des opérations.
- **Adaptabilité**: L'interface doit s'adapter aux besoins et aux préférences de l'utilisateur, en offrant de la flexibilité et en prenant en compte son expérience.
- **Gestion des erreurs**: L'interface doit aider l'utilisateur à prévenir, détecter et corriger les erreurs, avec des messages clairs et des mécanismes de récupération.
- **Homogénéité/cohérence**: L'interface doit être cohérente en interne (au sein de l'application) et en externe (avec d'autres applications), en utilisant des conventions de conception standard.
- **Signifiance des codes et dénominations**: Les codes et les termes utilisés dans l'interface doivent être significatifs et compréhensibles pour l'utilisateur.
- **Compatibilité**: L'interface doit être compatible avec les caractéristiques de l'utilisateur (âge, compétences, etc.) et avec les tâches qu'il doit effectuer.

Les 7 règles d'or de Coutaz

Joëlle Coutaz, chercheuse en IHM, a formulé sept règles d'or qui mettent l'accent sur la cohérence, la simplicité, la réduction de la charge cognitive, le contrôle utilisateur, la flexibilité, la structuration du dialogue et la prévention des erreurs.

- **Lutter pour la cohérence:** Assurer la cohérence interne (au sein de l'application) et externe (avec d'autres applications) est primordial. Cela signifie utiliser des éléments d'interface, des comportements et des terminologies de manière uniforme et prévisible. La cohérence facilite l'apprentissage et réduit la confusion de l'utilisateur.
- **Lutter pour la concision**: L'interface doit être concise et éviter les informations superflues. Les messages, les étiquettes et les instructions doivent être clairs, brefs et pertinents. Une interface concise permet à l'utilisateur de se concentrer sur les tâches essentielles.
- **Réduire la charge cognitive**: Minimiser l'effort mental requis de la part de l'utilisateur est crucial. Cela implique de présenter l'information de manière claire et structurée, d'éviter les distractions inutiles et de fournir des aides à la navigation et à la compréhension.
- **Mettre le contrôle entre les mains de l'utilisateur**: L'utilisateur doit avoir le sentiment de contrôler l'interaction. Cela signifie lui donner la possibilité d'initier des actions, d'annuler des opérations, de personnaliser l'interface et de naviguer librement.
- **Souplesse d'utilisation**: L'interface doit être flexible et permettre à l'utilisateur d'accomplir ses tâches de différentes manières. Offrir des raccourcis clavier, des menus contextuels et des options de personnalisation contribue à la souplesse d'utilisation.
- **Structurer le dialogue**: L'interaction doit être structurée de manière logique et intuitive. Les étapes d'une tâche doivent s'enchaîner de manière cohérente, et l'utilisateur doit pouvoir comprendre facilement où il se trouve dans le processus.
- **Prédire les erreurs**: Anticiper les erreurs potentielles de l'utilisateur et mettre en place des mécanismes pour les prévenir ou les corriger est essentiel. Cela peut inclure des messages d'erreur clairs, des confirmations avant les actions irréversibles et des options de récupération en cas d'erreur.

Modèles et théories

Représentations visuelles et théorie de Bertin

- **Variables visuelles** : Ce sont les différents éléments visuels (position, taille, forme, orientation, couleur, texture, intensité, transparence, etc.) qui peuvent être manipulés pour créer des représentations efficaces.
- **Théorie de Bertin** : C'est un cadre pour comprendre comment les variables visuelles peuvent être utilisées pour représenter des données de manière claire et significative. Elle distingue les variables planaires (x, y) et les variables rétiniennes (taille, forme, orientation, couleur, texture, luminosité), et met en évidence leurs propriétés :
    - **Associatives** : Permettent d'associer un élément à un groupe.
    - **Sélectives** : Permettent de différencier un élément d'un groupe.

Traitement pré-attentif :

Il s'agit du traitement visuel rapide (moins de 200-250 ms) et inconscient de l'information.

Exemple : la capacité à détecter rapidement une forme ou une couleur différente dans un ensemble.

Cécité au changement :

C'est la difficulté à percevoir des changements visuels lorsqu'ils se produisent en même temps qu'une interruption visuelle (par exemple, un clignotement ou un mouvement).

Théorie de la Gestalt

Elle étudie comment les humains perçoivent et organisent les éléments visuels en groupes ou en formes.

Lois de la Gestalt :

- **Proximité** : Les éléments proches sont perçus comme appartenant à un même groupe.
- **Similarité** : Les éléments similaires sont perçus comme appartenant à un même groupe.
- **Continuité** : Les éléments qui forment une ligne ou une courbe continue sont perçus comme appartenant à un même groupe.
- **Fermeture** : Les éléments qui forment une figure fermée sont perçus comme un tout.
- **Expérience** : Les éléments familiers ou significatifs sont plus facilement perçus.

Modèles prédictifs de performance

- **Loi de Fitts** : Prédit le temps requis pour atteindre une cible à l'écran en fonction de sa taille et de sa distance.
- **Loi de Hick-Hyman** : Prédit le temps de réaction en fonction du nombre de choix possibles.

Capacités humaines

- Temps de réaffichage : Un réaffichage inférieur à 1/10 de seconde crée un effet d'animation.
- Vitesse maximale de la main : 1 à 1,5 m/s pour le pointage.
- Le nombre magique 7 \u00b1 2 : Nombre d'éléments qu'une personne peut retenir en mémoire à court terme.

Modèles perceptuel et conceptuel

- **Modèle perceptuel** : Modèle mental construit par l'utilisateur à partir de ses perceptions et de son expérience.
- **Modèle conceptuel** : Description et fonctionnement du système tel que conçu par les développeurs.
- **Performance (utilisabilité)** : Déterminée par la distance entre le modèle perceptuel et le modèle conceptuel.

Keystroke-Level Model (KLM)

Modèle pour prédire le temps d'exécution d'une tâche en la décomposant en actions élémentaires (frappe de touche, pointage, déplacement de la main, activité mentale, etc.).

Chaque action élémentaire a un temps d'exécution estimé.

Permet de comparer l'efficacité de différentes méthodes pour réaliser une tâche.