**Prinsip Desain Fundamental Donald Norman: Meminimalisir Kebingungan Pengguna**

Donald Norman, yang dikenal sebagai salah satu pendiri HCI (Human-Computer Interaction) dan penemu istilah UX, menekankan pentingnya Desain yang Baik (*Good Design*) untuk mengatasi kebingungan yang dialami pengguna saat berinteraksi dengan produk atau sistem interaktif. Kebingungan ini muncul dari dua "jurang" utama:

* **Jurang Eksekusi (*Gulf of Execution*)**
  + **Penjelasan:** Ini adalah kesenjangan antara apa yang diinginkan atau dipikirkan pengguna dengan apa yang disediakan atau diizinkan oleh aplikasi atau sistem interaktif1. Pengguna merasa tidak tahu apa yang bisa mereka lakukan2.
  + **Contoh:** Bayangkan kamu baru pertama kali menggunakan mesin kopi otomatis di kafe. Ada banyak tombol dengan ikon-ikon yang tidak jelas, dan tidak ada petunjuk sama sekali. Kamu tidak tahu tombol mana untuk espresso, mana untuk latte, atau bagaimana cara memulai prosesnya. Ini adalah *Gulf of Execution* karena kamu tidak tahu apa yang bisa kamu lakukan dengan mesin itu.
* **Jurang Evaluasi (*Gulf of Evaluation*)**
  + **Penjelasan:** Ini adalah kesenjangan antara ekspektasi pengguna dengan reaksi atau *output* yang diberikan oleh aplikasi atau sistem interaktif3. Pengguna merasa tidak tahu apa yang terjadi setelah melakukan tindakan4.
  + **Contoh:** Kamu menekan tombol pada mesin kopi tadi, tapi tidak ada lampu yang menyala, tidak ada suara, dan tidak ada kopi yang keluar. Kamu menunggu beberapa detik, tapi tetap tidak ada reaksi. Kamu bingung, apakah tombolnya rusak? Apakah sudah dipencet atau belum? Ini adalah *Gulf of Evaluation* karena kamu tidak tahu apa yang terjadi atau apakah tindakanmu berhasil.

**Pentingnya *Good Design*: Kemampuan Menemukan (*Discoverability*) + Pemahaman (*Understanding*)** 555

Untuk meminimalisir kedua jurang ini dan menciptakan

*Good Design*, Norman menekankan dua hal utama: *Discoverability* dan *Understanding*6.

* **Kemampuan Menemukan (*Discoverability*)** 77
  + **Penjelasan:** Pengguna harus mudah menemukan apa saja yang bisa mereka lakukan, di mana mereka bisa melakukannya, dan bagaimana melakukannya8888. Ini juga berarti pengguna bisa menemukan atau merasakan reaksi dari suatu tindakan9. Prioritaskan bahwa fitur atau

*output* yang paling penting berarti harus yang paling mudah ditemukan10.

* + **Contoh Baik:** Halaman utama https://www.google.com/search?q=Google.com. Fitur dasarnya, yaitu kolom pencarian, sangat jelas terlihat di tengah halaman, sehingga sangat mudah ditemukan oleh siapa saja11. Ini menunjukkan

*Discoverability* yang baik karena yang paling penting (pencarian) adalah yang paling mudah ditemukan.

* + **Contoh Buruk:** Tombol *Power* di Windows 8. Bagi pengguna lama Windows yang terbiasa menemukan tombol *Power* di sudut kiri bawah layar, memindahkannya ke sudut kanan bawah di tampilan yang berbeda membuat *Discoverability*-nya buruk karena fitur dasar menjadi sulit ditemukan12.
* **Pemahaman (*Understanding*)** 1313
  + **Penjelasan:** Tujuannya adalah membantu pengguna memahami apa yang mereka lakukan atau apa yang sedang terjadi141414. Menemukan sesuatu tidak berarti memahami sesuatu15.
  + **Contoh:** Kamu menemukan sebuah ikon berbentuk amplop di aplikasi email. Dari bentuknya, kamu *memahami* bahwa ikon itu mungkin berkaitan dengan email atau pesan. Tanpa pemahaman ini, meskipun kamu menemukannya (*Discoverability*), kamu tidak akan tahu apa fungsinya.

**Tujuh Prinsip Desain Fundamental untuk Meningkatkan *Discoverability* dan *Understanding*** 16

1. **Model Konseptual (*Conceptual Model*)** 17
   * **Penjelasan:** Membantu pembentukan *Mental Model* pengguna yang benar tentang cara kerja sistem18181818.

*Mental Model* adalah model dalam pikiran seseorang yang merepresentasikan pemahaman mereka tentang bagaimana sebuah produk bekerja atau berfungsi19. Meskipun

*Mental Model* biasanya sederhana, ia harus menyerupai cara kerja sistem yang sebenarnya20.

*System Image* adalah bentuk penyederhanaan dari *Conceptual Model* yang nantinya ditampilkan kepada pengguna (misalnya, UI)21. Penting untuk diingat bahwa

*Conceptual Model* boleh rumit, tetapi *System Image* harus dicoba dibuat sesederhana mungkin agar *Mental Model* pengguna mudah terbentuk222222222222222222.

* + **Contoh:** Ketika kamu mengendarai mobil, *Mental Model* sederhanamu adalah: injak gas untuk bergerak, injak rem untuk melambat, dan putar setir untuk berbelok23. Kamu tidak perlu memahami secara detail cara kerja mesin, transmisi, atau sistem pengereman yang rumit di dalamnya24.

*System Image* (UI mobil seperti pedal dan setir) yang sederhana ini membantu membentuk *Mental Model* yang benar dan cukup untuk mengoperasikan mobil25.

1. **Affordance** 26
   * **Penjelasan:** *Affordance* adalah interaksi yang mungkin terjadi antara seseorang dengan suatu objek27. Ini adalah relasi antara benda/objek fisik dengan manusia/makhluk hidup yang membolehkan mereka melakukan suatu aksi menggunakan/terhadap objek tersebut28.

*Affordance* muncul karena pengguna menangkap atribut atau properti dari objek dan memikirkan apa saja yang mungkin mereka lakukan dengan atribut atau properti tersebut29. Dalam UX Design, untuk fitur-fitur yang penting, desainer perlu memastikan untuk tidak terjadi

*Hidden Affordance*30. Desainer juga perlu meminimalisir jumlah

*False Affordance* karena ini adalah konsep *Gulf of Execution*31.

* + **Jenis-jenis Affordance:** 32
    - ***Actual Affordance:*** *Affordance* yang sebenarnya dimiliki objek33.
      * **Contoh:** Sebuah gelas plastik *benar-benar* bisa diisi air34.
    - ***Perceived Affordance:*** *Affordance* yang orang pikir/percaya dimiliki oleh objek35.
      * **Contoh:** Sebuah pintu dengan pegangan horizontal (model batang dorong) akan membuatmu *berpikir* bahwa pintu itu harus didorong36.
    - ***Hidden Affordance:*** *Affordance* yang dimiliki benda tapi orang tidak menyadarinya37.
      * **Contoh:** Lubang kecil pada kaleng soda (di atas *pull tab*) yang sebenarnya bisa digunakan untuk menahan sedotan agar tidak tenggelam ke dalam minuman38. Banyak orang tidak menyadari

*affordance* ini.

* + - ***False Affordance:*** Konflik antara *Perceived Affordance* dan *Actual Affordance*39.
      * **Contoh:** Pintu yang memiliki pegangan tarik, tetapi sebenarnya hanya bisa didorong. Kamu mencoba menariknya (

*Perceived Affordance*), tetapi tidak terbuka, karena *Actual Affordance*-nya adalah didorong40.

1. **Penanda (*Signifier*)** 41414141
   * **Penjelasan:** *Signifier* adalah segala petunjuk yang dapat "ditangkap" manusia (*perceive*) yang mengkomunikasikan *affordance* objek yang benar.

*Signifier* berfungsi untuk memberi petunjuk kepada pengguna agar menggunakan objek dengan benar42424242. Petunjuk yang diberikan

*Signifier* adalah petunjuk: Apa, Di mana, dan Bagaimana. *Signifier* bisa berupa teks, simbol/gambar/ikon, warna, suara, lampu, bayangan, getaran, atau apapun. Dengan memberikan

*Signifier*, diharapkan dapat mempermudah *Mental Model*, meningkatkan *Discoverability*, serta meminimalisir *Hidden Affordance* dan *False Affordance*43.

* + **Contoh Apa (Fungsi):** Tulisan "EXIT" pada pintu keluar kebakaran44.
  + **Contoh Di Mana (Lokasi):** Tanda panah di peta navigasi GPS yang menunjukkan arah belok45.
  + **Contoh Bagaimana (Cara Penggunaan):** Tulisan "DORONG" atau "TARIK" pada sebuah pintu46. Atau tulisan "PRESS START" dan "INSERT COIN TO CONTINUE" di game arcade memberi petunjuk bagaimana cara memulai game47.
  + **Contoh Bentuk *Signifier* Non-Teks:**
    - **Ikon:** Ikon tempat sampah untuk fungsi "delete"48.
    - **Warna:** Warna merah pada tulisan error yang langsung menandakan ada yang salah49.

1. **Batasan (*Constraint*)**
   * **Penjelasan:** *Signifier* hanya bisa memberi petunjuk, ia tidak benar-benar bisa mencegah. *Constraint* berfungsi untuk menghalangi atau membatasi interaksi pengguna. Dengan *Constraint*, kita bisa meminimalisir pengguna melakukan tindakan yang salah.
   * **Contoh:** Bentuk port USB pada komputer. *Port* ini didesain sedemikian rupa sehingga *plug* USB hanya bisa masuk dengan satu orientasi yang benar. Ini adalah

*constraint* fisik yang mencegah pengguna melakukan kesalahan saat mencolokkan perangkat50.

* + **Contoh dalam UI:** Pada formulir pendaftaran, jika ada pilihan jenis kelamin, penggunaan *radio button* (misalnya: "Laki-laki" atau "Perempuan") adalah *constraint* yang memastikan pengguna hanya bisa memilih salah satu, bukan mengetik sembarang teks51. Sama juga dengan pemilihan tanggal lahir yang menggunakan

*dropdown* bukan *textbox*52.

* + **Contoh Alur Proses:** Dalam proses *checkout* belanja *online* yang memiliki langkah-langkah 1-6, kamu tidak bisa langsung loncat ke langkah terakhir tanpa menyelesaikan langkah-langkah sebelumnya53. Ini adalah

*constraint* proses yang membatasi interaksi pengguna untuk memastikan alur yang benar.

* + Kombinasi antara *Signifier* dan *Constraint* akan membantu meminimalisir *Gulf of Execution*.

1. **Umpan Balik (*Feedback*)**
   * **Penjelasan:** *Feedback* mengkomunikasikan hasil suatu tindakan. *Feedback* yang baik akan membantu pengguna membentuk *Mental Model*-nya, yang ujung-ujungnya adalah meningkatkan *Understanding*.
   * **Ciri-ciri *Feedback* yang baik:** Langsung (reaksi tanpa *delay*) 54, Menunjukkan kondisi aplikasi/sistem interaktif yang sebenarnya (tidak berbohong atau salah) 55, Menunjukkan jika tindakan

*user* benar/salah 56, Bisa ditangkap oleh

*user* 57, Bisa dipahami oleh

*user*58.

* + *Feedback* dapat dibedakan menjadi tiga jenis:
    - ***Positive Feedback:*** *Feedback* sebagai reaksi terhadap tindakan *user* yang benar.
      * **Contoh:** Setelah memasukkan *username* dan *password* yang benar, kamu langsung dialihkan ke halaman beranda.
    - ***Negative Feedback:*** *Feedback* sebagai reaksi terhadap tindakan *user* yang salah. Harus memenuhi 3 syarat: tahu ada yang salah 59, tahu kenapa salah 60, dan tahu bagaimana cara memperbaikinya61.
      * **Contoh:** Saat kamu mencoba *login* ke Facebook dengan email yang salah, muncul pesan "Incorrect Email" dengan latar belakang merah, menjelaskan bahwa email tidak terdaftar 62dan menyarankan untuk memastikan pengetikan yang benar ("Make sure that it is typed correctly")63.
    - ***Progress Feedback:*** *Feedback* sebagai informasi bahwa proses belum selesai.
      * **Contoh:** Ketika kamu memindahkan file berukuran besar di komputer, muncul *progress bar* yang menunjukkan berapa persen file yang sudah dipindahkan64.
  + *Feedback* yang baik sangat penting untuk menutup *Gulf of Evaluation* 65dan mengembangkan

*Mental Model* pengguna66.

1. **Pemetaan (*Mapping*)** 67
   * **Penjelasan:** *Mapping* adalah relasi antara *remote control* atau *controller* dengan objek yang dikontrol68686868. Penting untuk memiliki

*mapping* yang baik agar *Mental Model* dari objek yang dikontrol mudah dipahami69696969. Tujuannya adalah agar relasi dapat ditunjukkan tanpa perlu banyak menggunakan

*Signifier*, cukup dengan menggunakan bentuk dan letak70707070.

* + **Contoh:** Tata letak tombol pada kompor empat tungku yang diatur sesuai dengan posisi fisik tungku di atasnya (misalnya, tombol kiri atas mengontrol tungku kiri atas, dan seterusnya)71. Ini adalah

*mapping* yang baik karena relasi antara kontrol dan objek yang dikontrol sangat intuitif berdasarkan lokasi, tanpa perlu label tambahan72.

* + **Contoh dalam Game:** Penggunaan tombol WASD pada *keyboard* untuk pergerakan dalam *game*. Posisi W (atas), A (kiri), S (bawah), D (kanan) secara alami merepresentasikan arah, sehingga mudah dipahami tanpa perlu tulisan "maju", "mundur", "kiri", "kanan" di setiap tombol73.

Semoga penjelasan yang terintegrasi dengan contoh ini membuat setiap konsep lebih mudah kamu pahami dan ingat untuk ujian!