PERKULIAHAN KE 7

Tujuan Instruksional Khusus (TIK)

- Mengenali teknik pemodelan, model user dan kognitif
- Menjelaskan model-model interaksi
- Menyebutkan bagaimana teknik semi formal yaitu analisa status/ kejadian.

Pokok Bahasan : Formalisasi Standard, Model-Model Interaksi, Analisa Status/ Kejadian.

Deskripsi Singkat : bahasan ini tentang cara memodelkan semantik dari sebuah sistem yang interaktif. Dialog adalah tentang bagaimana aksi user yang sesuai berhubungan dengan sistem.

Bahan Bacaan : Dix, Alan et.al, *HUMAN-COMPUTER INTERACTION*, Prentice Hall, Europe, 1993, hal 299-334

MODEL-MODEL SISTEM

Tinjauan

1. Formalisasi standard dari RPL untuk membuat sistem yang interaktif, yaitu berbasis model, misal Z, formalisasi aljabar, logika temporal dan deontic
2. Model-model interaksi khusus, seperti predictability dan observability; reachability dan undo
3. Model interaksi yang lengkap, yang dapat mengelompokkan fenomena seperti event atau status, memasukkan psikologi dan penekanan masalah feedback pada interface.

Formalisasi Standard

Tujuan spesifikasi formal ada 2 :

- Komunikasi
- Analisa

Notasi formal untuk komunikasi

Spesifikasi dapat dibuat sebagai bahasa yang 'umum' antar tim desain, desainer dan pembuat sistem. Ide2 tentang tampilan layar dapat dengan muda divisualisasikan dengan bantuan paket2 untuk menggambar (*drawing tool*), tetapi perilaku sistem yang dinamis sulit dikomunikasikan.

Sering spesifikasi formal menjadi ambigu sehingga deskripsi sistem menjadi ambigu pula ⟵ salah !

Simbol yang digunakan dan manipulasinya mempunyai arti dalam sistem formal, tetapi interpretasi simbol tersebut dapat berbeda untuk setiap orang.

Contoh : layout layar untuk warna (x,y) koordinat (0,0)

(x,y) = (0,0) = ? (kiri bawah atau kanan bawah)
ambigu

PENTING !, jika membuat spesifikasi formal perlu disertai dengan penjelasan/ komentar dan deskripsinya/ dokumentasi.

Notasi Formal untuk analisa

Spesifikasi formal dapat dianalisa dalam berbagai cara :

- Periksa konsistensi internal,

Lihat jika setiap statement dibuat dalam satu bagian yang saling berkontradiksi. Contoh : umumnya tidak memungkinkan membangun system yang sesuai dengan properti antara teori dan praktek.

- Periksa konsistensi eksternal,

Yang berhubugan dengan program (bukan keuntungan dalam IMK). Tugas verifikasi ini merupakan keuntungan dari spesifikasi formal dari sudut pandang RPL.

- Periksa konsistensi eksternal,

Yang berhubungan dengan kebutuhan2, beberapa diantaranya seperti properti keamanan, sistem khusus, dll. Kebutuhan lainnya tentang system khusus misalnya fungsi yang dapat diakses hanya dengan penekanan kurang dari 3 keystrokes.

Notasi Berorientasi Model

Dimulai pada akhir 1970 dan 1980 untuk menyediakan software engineer kemampuan menggambarkan dan alasan tentang komponen software yang menggunakan konstruksi matematika seperti konstruksi yang digunakan dalam bahasa pemrograman. Notasi matematika ini menggambarkan perilaku dari system software yang didekatkan dengan bagaimana diprogramkan.

Ada 2 yang digunakan : Z dan VDM, digunakan untuk spesifikasi interface. Z digunakan untuk menspesifikasikan editor, window manager dan toolkit grafik Presenter.

❖ Simple Sets

Set yang paling sederhana (standard) : R = bilangan real, Z = bilangan integer, N = bilangan asli. Yang non standard didefinisikan sebagai set baru dengan melisting angka-angka finite dari nilai yang mungkin dari set tersebut. Misal bentuk2 geometri dalam grafik.

Shape_type ::= line | ellipse | rectangle
Keystrokes ::= a | b | … | z | A | B | … | 0 | … | 9 | cursor_left | …
Atau [Keystrokes]

Set2 tersebut dapat dibuat lebih kompleks lagi, meliputi tupel yang terurut, tidak terurut, yang disebut skema dalam Z, deretan dan fungsi.

Contoh : koordinat (x,y) untuk titik (point) memerlukan 2 tupel (pasangan terurut) dari R :

Point ::= R x R; misal Point (1.2, -3.0)

Bentuk geometri dalam 4 tupel :

Shape_type x R x R x Point

Skema Z :

Shape
type : Shape_type
wid : R
ht : R
centre : Point

Jika menggunakan deklarasi skema, shape s dengan width atau centre dari s dituliskan sebagai s.wid atau s.centre (seperti bahasa pascal dengan tipe record, C dengan ‘struct’)

Deretan (sequence) dapat mempunyai panjang tetap seperti array pada pascal. Dalam matematika panjang deretan bisa bervariasi. Dua deretan a dan b dapat digabungkan dengan memberikan deretan baru, ditulis sebagai a⌒b – tipe list dalam LISP.

Function : perhitungan standard dalam bahasa pemrograman. Fungsi memetakan elemen2nya dari satu himpunan ke himpunan lainnya dan berlaku juga sebagai ‘kamus lookup’. Contoh : sqrt atau log. Bergantung pada konteks yang digunakan, fungsi dalam spesifikasi diimplementasikan dengan fungsi tingkat program atau struktur data. Contohnya adalah fungsi untuk grafik, misal

[Id]
Shape_dict == Id $\rightarrow\!\!\!\!\!+$ Shape

Set Id adalah set dari identifier yang akan digunakan sebagai label dari shape. Objek shapes dari tipe shape_dict adalah label yang memetakan fungsi ke shapes. Jika id adalah label khusus maka kamus shapes dapat memetakan ke persegi panjang-rectangle dengan lebar 2.3, tinggi 1.4 dan pusat (1.2, -3), dituliskan sebagai :

Shape(id).type = rectangle
Shape(id).wid = 2.3
Shape(id).height = 1.4
Shape(id).centre = (1.2, -3.0)

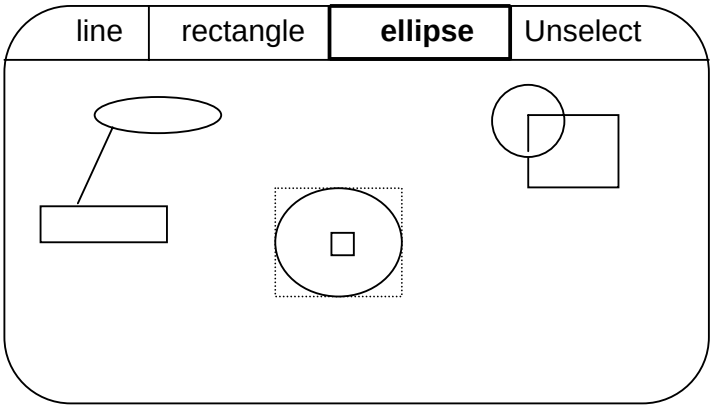
Shape_dict sebagai fungsi parsial. Fungsi parsial tidak dapat memetakan semua elemen sumber ke elemen himpunan tujuan. Tidak semua dalam Id adalah argument yang valid untuk shapes. Himpunan nilai yang valid disebut domain dari shapes 'dom shapes'

dom shapes = {5,1,7,4}

shapes(5),shapes(1),shapes(7),shapes(4) adalah valid.

❖ Zdraw – state dan invariant (sebuah system grafik sederhana)

Spesifikasi orientasi model sering ditulis dalam model yang imperative. Satu mendefinisikan state dari system dan kemudian operator sebagai efeknya. State dan operator dalam Z ditulis dalam notasi skema.



SKEMA

State shapes : shape_dict selection : P id selection $\subseteq$ dom shapes	identifikasi/ definisi dari komponen sistem grafik
Init State dom shapes = { } selection = { }	Sistem model geometri. State tanpa bentuk (shapes) yang dibuat/ dipilih
Init State dom shapes = { }	Definisi dari state awal yang menghilangkan predikat

Skema State terbagi dalam 2 bagian, yang ditandai dengan garis. Di atas garis sebagai komponen dari state system grafik. Di bawah garis adalah invariant – kondisi yang harus dipenuhi komponen state. Predikat pada garis yang terpisah diasumsikan digabungkan dengan logika “and”. Set dengan objek yang terpilih sekarang harus selalu berada dalam set dari objek yang dibuat pada system. Kita tidak diperkenankan memilih objek yang lain yang tidak dibuat user. System pemodelan geometri ini dimulai dengan state Init.

❖ Mendefinisikan Operasi

Untuk mendefinisikan operasi, perlu dibuat state dari sistem grafik. Misal untuk membuat shape baru seperti lingkaran, kita harus memilih ellipse. Lingkaran berukuran tetap muncul di layer, kemudian user bebas memindah dan membentuk ke posisi yang diinginkan.

Untuk mendefinisikan sebuah operasi, gambaran dari state system grafik sebelum dan sesudah operasi dibuat. Misal

State $\longrightarrow$ sebelum kopi ? $\rightarrow$ input
State’ $\longrightarrow$ sesudah kopi ! $\rightarrow$ output

Membuat New Ellipse yang berukuran tetap (diasumsikan berpusat di (0,0)).

NewEllipse
State
State’
newid? : Id
newshape? : Shape
---
newid? $\notin$ dom shapes
newshapes?.type = Ellipse
newshapes?.wid = 1
newshapes?.ht = 1
newshapes?.centre = (0,0)
shapes’ = shapes U {newid? $\longrightarrow$ Newshapes?}
selection’ = {newid?}

Hasil operasi ini adalah ter-update-nya kamus shape karena ada identifier ‘fresh’ ke bentuk elips baru yang didefinisikan dengan newshape?. Objek yang terpilih adalah objek baru.

Unselect	Bentuk akan tetap sama setelah
State	operasi (secara eksplisit)
State’	
Selection’ = { }	Operasi Unselect menjadikan
Shapes’ = shapes	objek yang dipilih KOSONG

Issue for model-oriented notations

‘Framing problems’, terlalu mengerti tentang sesuatu yang akan terjadi secara eksplisit, tetapi sulit untuk memformulasikan (jika sesuatu tidak disebutkan, diasumsikan tidak berubah). Misal pada state inisialisasi Init, tidak disebutkan secara eksplisit bahwa bentuk yang dipilih adalah kosong. Pada operasi Unselect sangat penting secara eksplisit disebutkan bahwa kamus shape tetap sama setelah operasi.

‘Separation’, antara fungsionalitas sistem dan presentasi. Dalam contoh sebelumnya, ada identifikasi kamus bentuk yang dibuat, tetapi tidak disebutkan bagaimana bentuk2 tersebut dipresentasikan ke layar user. Informasi presentasi tidak diperlukan untuk mendefinisikan bagaimana system bekerja secara internal, tetapi penting untuk tepat tentang isu presentasi tentang fungsionalitas system.

Notasi Aljabar
Pada prinsipnya, spesifikasi aljabar tidak membuat gambaran dari komponen sebuah objek tetapi lebih ke seperti apa gambaran objek terlihat dari luar.

Aljabar versi Zdraw (lihat contoh sistem grafik sebelumnya).
Operasi ‘select’ : memilih objek terdekat dari pilihan yang ada
Operasi ‘unselect’ : menghapus pilihan yang ada.

Spesifikasi aljabar tidak menyediakan representasi/ model eksplisit dari sistem. Tipe-tipe yang penting dari state dideklarasikan dengan set operasi yang memanipulasinya. Set dari aksioma kemudian secara implicit mendefinisikan state system.

Algebraic-draw =
types
State, Pt
operations
init : $\longrightarrow$ state
new_ellipse, new_rectangle, new_line : Pt x State $\longrightarrow$ State
move : Pt x State $\longrightarrow$ State
unselect : State $\longrightarrow$ State
delete : State $\longrightarrow$ State
axioms
for all st $\in$State; p, p’ $\in$ Pt •
1. delete(new_ellipse(st)) = unselect(st)
2. delete(new_rectangle(st)) = unselect(st)
3. delete(new_line(st)) = unselect(st)
4. move(p, unselect(st)) = unselect(st)
5. resize(p, unselect(st) = unselect(st)
6. move(p, move(p’,st)) = move(p, st)
7. resize(p, resize(p’,st)) = resize(p, st)
8. delete(delete(st)) = delete(st)

Pertama mendeklarasikan tipe penting di spesifikasi yang akan bertindak sebagai argument pada beberapa operasi, dan state system grafik secara keseluruhan. Tidak ada informasi tersedia tentang bagaimana konstruksi dari tipe-tipe ini. Operasi dibuat dan didefinisikan sebagai input dan output. Operasi new_ellipse didefinisikan untuk satu titik dan current state sebagai argument dan mengembalikan state baru. Operasi init tidak mengambil argument dan menghasilkan satu state. Tipe dan operasi bersama2 membentuk signature dari spesifikasi AlgebraicDraw.

Aksioma adalah keterhubungan antara bermacam2 operasi. Aksioma menunjukkan bagaimana operasi berinteraksi dengan lainnya. 3 aksioma pertama merupakan pembuatan berbagai objek dan kemudian menghapusnya ini tidak mempunyai efek

selain tidak memilih current objek. 2 aksioma berikutnya untuk memindah dan membentuk ulang ketika tidak ada objek yang dipilih sama sekali. Aksioma 6 dan 7, memindah dan membentuk ulang terlupakan jika kita melakukan dua pembentukan ulang pada dua baris sebelumnya seolah-olah kejadian pertama tidak pernah terjadi. Pada pendekatan berorientasi model, akan lebih mudah menspesifikasikan bahwa operasi perpindahan adalah kumulatif – dua perpindahan sebelumnya adalah sama dengan melakukan satu pemindahan yang sama dengan jumlah dari dua pemindahan. Aksioma yang terakhir adalah aksi menghapus yang semakin terlupakan karena idempotent – melakukan penghapusan yang kedua adalah sia-sia.

Membaca urutan dalam formula aljabar

resize(p, move(p', new_rectangle(st)), berarti melakukan new_rectangle, kemudian melakukan pemindahan dan kemudian pembentukan ulang. Akan lebih jelas jika dituliskan sebagai

st_1 = new_rectangle(st)
st_2 = move(p', st_1)
st_3 = resize(p, st_2)

axiom 1 dan 5 dapat dituliskan

new_ellipse; delete = unselect
unselect; resize(p) = unselect

Logika Temporal dan Lainnya

Huruf digunakan untuk merepresentasikan statemen logika. Misal

$(p \wedge q) \vee r$, dimana
p = 'my nose is green'
q = 'I've got ears like a donkey'
r = 'I'm called Alan'

Logika predikat dan logika proporsional ($P(x) \vee Q(x)$) digunakan sebagai bagian dari formalisasi. Kedua logika ini merupakan keluarga logika yang sangat sederhana yang dibentuk sebagai bagian dari filosofi dan logika matematika.

Simbol-simbol pada logika temporal.
Simbol dasar adalah $\Box$ $\Diamond$ $\forall$ $\exists$ $\neg$ (always, eventually, for all, there exist, not)
$\Box\neg$ = never
contoh :

- $\Box$(rains on Tuesday) = "always rains on Tuesday"
- $\Box(\neg p)$ it is always <u>not true</u> when <u>p</u> = p never happens
- $\Box\neg$(computer explodes)
- $\Box$(user types 'print fred' $\Rightarrow$ $\Diamond$ the laser printer prints the file 'fred')
  = at all time, if the user types the command 'print fred', then eventually the file 'fred' will be printed on the laser printer.

Operator tambahan
Eventually $\longrightarrow$ before the end of this interval
p until q – p harus benar <u>sampai q</u> benar
p before q – p harus benar pada beberapa saat <u>sebelum q</u> benar

p until q, p benar hingga q benar lebih lemah dari □p, p benar selamanya
p before q lebih kuat dari ◇p.

(◇ user types ‘print fred’) ⇒ the laser printer prints the file ‘fred’
= if at any time in the future the user types ‘print fred’, then the system ought to print the file ‘fred’ now.

Logika Deontic

Logika deontic meliputi konsep agent (human, corporate dan computer) yang bertanggungjawab dan saling ketergantungan di antara agent.
Operator umum : permission (per), obligation(obl). Kedua operator ini menggunakan dua argument; pertama adalah siapa yang memiliki permission atau obligation dan yang kedua adalah apakah mereka diijinkan atau diwajibkan untuk dibuat benar.
Contoh : perbaikan logika temporal sebelumnya,

Misal agent user ⟶ ‘Jane’
agent laser printer ⟶ ‘lp3’

owns(Jane,file ‘fred’) ⇒ per(Jane, request(Jane,’print fred’))

Clumsy

type(Jane,’print fred’) ⇒ obl(lp3, prints the file ‘fred’)

Pada formula yang pertama, jika Jane memiliki file ‘fred’ maka ia dijinkan untuk meminta perintah ‘print fred’. Di formula yang kedua disebutkan bahwa jika ia meminta perintah maka printer diwajibkan untuk mencetak ‘fred’.

request(‘print fred’), dapat dilakukan oleh agent sehingga formula di atas dapat diperbaiki sebagai :

owns(Jane, file ‘fred’) ⇒ per(Jane, request(Jane,’print fred’))
performs(Jane, request(‘print fred’)) ⇒ obl(lp3, print(the file ‘fred’))

Model-Model Interaksi

WYSIWYG – What You See Is What You Get

System yang interaktif memiliki istilah ini, konsisten, memiliki fasilitas undo yang universal, dan lain-lain.

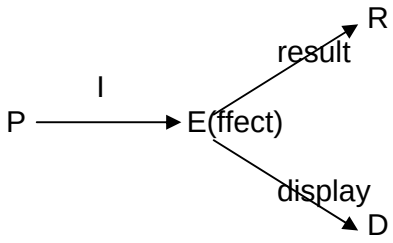
Model PIE

Model PIE adalah model black box, tidak untuk representasi arsitektur internal dan konstruksi dari system computer, tetapi menggambarkan input dari user dan output ke user. Perbedaan antara display sesaat dari system dan hasil yang permanent adalah sentral dari model PIE. Display disimbolkan D dan hasil disimbolkan dengan R. Prinsip observability merupakan hubungan antara display dan hasil.

Statement formal dari predictability adalah tentang state internal dari system. Ini bukan model black box. Pertama, state akan opaque-buram, tidak akan terlihat strukturnya hanya dalilnya. Kedua, state sebenarnya tidak akan dibicarakan hanya sifat idealnya. Akan merupakan state minimal bagi bakal perilaku eksternal, ini yang disebut sebagai efek (E).

Display : E ⟶ D, result : E ⟶ R; fungsi interpretasi, I : P ⟶ E

Aksi user memberi perintah (commands = C) biasa disebut Program. Semua sejarah dari perintah user disebut program (P = seqC) dan efek current dikalkulasikan dari sejarah dengan menggunakan fungsi interpretasi (I).



Fungsi transisi state-nya :

doit : E x P ⟶ E

fungsi doit mengambil present state e dan beberapa perintah user p, dan memberikan stat baru setelah user memasukkan perintah doit(e,p).

Fungsi interpretasi(I) berrelasi dengan aksioma berikut :

doit(I(p),q) = I(p⌒q)
doit(doit(e,p),q) = doit(e, p⌒q)

Diagram PIE dapat dibaca pada level abstraksi yang berbeda.

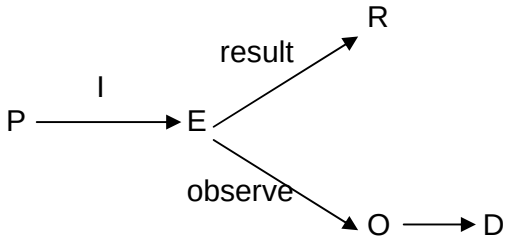
C : keystrokes atau klik mouse
C = {'a', 'b', …, '0', '1', …, '*', '&', …}
D : display fisik
D = pixel_coord ⟶ RGB_value
R : output yang dicetak
R = ink on paper

Model PIE merupakan interpretasi pada level fisik/ leksikal. Lebih berguna lagi untuk mengaplikasikan model pada level logika.

'select bold font'
window, button, field, …

Model ini dapat diaplikasikan di banyak level abstraksi.

Predictability & Observability



WYSIWYG memiliki 2 interpretasi :

1. WYS adalah WYou will Get di printer, bagaimana kita menentukan hasil dari display

2. WYS adalah WYou have Got di sistem, apa yang display berikan tentang efek.
Ke-2 interpretasi ini dianggap sebagai prinsip observability.

Keadaan sistem menunjukkan efek dari perintah2 berikutnya, sehingga jika sistem dapat diobservasi, display-nya menunjukkan suatu keadaan yang berarti predictable. (kasus khusus dari observability).

Formalisasi property. Kita dapat menentukan hasil dari display jika ada fungsi transparansi dari display ke hasil :

$$\exists transparent_R : D \rightarrow R\bullet$$
$$\forall e \in E \bullet transparent_R (display(e)) = result(e)$$

Untuk efeknya, state systemnya :

$$\exists transparent_E : D \rightarrow E\bullet$$
$$\forall e \in E \bullet transparent_E (display(e)) = e$$

Efek observable (O)
Sistem akan result observable jika hasil dapat ditentukan dari efek observable :

$$\exists predict_R : O \rightarrow R\bullet$$
$$\forall e \in E \bullet predict_R (observe(e)) = result(e)$$

Efek observasi memuat minimal informasi sebanyak hasilnya. Juga berisi informasi tambahan dari keadaan interaksi sistem. Misalnya pada saat mau melakukan pencetakan.

Kondisi yang lebih kuat dari system yang fully predictable :

$$\exists transparent_E : O \longrightarrow E\bullet$$
$$\forall e \in E \bullet predict (observe(e)) = e$$

Kita dapat mengobserve state yang lengkap dari system. Kemudian kita dapat memprediksi apa pun dari system untuk melakukan sesuatu.

Reachability dan Undo
Suatu sistem dapat dikatakan tercapai (reachable) jika dari satu state dalam sistem dapat mencapai satu state lainnya.

$$\forall e, e' \in E \bullet (\exists p \in P \bullet doit(e,p) = e')$$

Contoh : memindah/ mengkopi dokumen antar layar.

Kasus khusus dari reachability : UNDO

$$\forall c \in C \bullet doit(e, c \frown undo) = e$$

Berawal dari state e. Kemudian kita lakukan satu perintah c dan diikuti dengan perintah khusus undo. Maka state nya akan sama seperti awalnya.

Model-model Interaksi Lainnya

- Windowing system
- Timing
- Attention
- Non determinism
- Dynamic pointers

Model PIE merupakan model event-in/ status-out. User melakukan event (dari C) dan system meresponnya dengan status (display).

Status / Event Analysis

Perbedaan status dan event adalah being dan doing. Status selalu memiliki nilai yang dapat direfer. Event merupakan kejadian pada saat tertentu. Analisis status event ini terlihat di layer system yang berbeda, user, layer (presentasi), dialog dan aplikasi. Pencarian event tercapai di setiap level dan status berubah pada setiap level. Pengggabungan dengan analisa psikologi yang naïf dari batasan presentasi membuat desainer memprediksi kesalahan dan perbaikan penting lainnya.

Properti event : waktu dan kalender.
Brian berjanji bertemu Alison untuk menonton bioskop jam 8 kurang 20. Dia berhenti bekerja 5 menit sebelumnya dan melihat ke jamnya terus menerus. Setiap beberapa menit, ia melihat jamnya. Pada saat jamnya menunjukkan jam 8 kurang 24 dia langsung pergi. Sebenarnya jam Brian memiliki alarm dan dia dapat melakukan setting jam 7:35 tetapi dia tidak tahu bagaimana cara melakukan settingnya.

Status : jamnya Brian, yang selalu menunjukkan waktu (dapat kontinyu maupun diskrit)

Event : waktu yang menunjukkan 7:35. Brian berhenti bekerja.

Polling : keadaan di mana Brian melihat ke jamnya terus menerus (periodic). Polling adalah kegiatan normal yang orang lakukan seperti mesin, merupakan cara standard untuk mengubah status menjadi event.

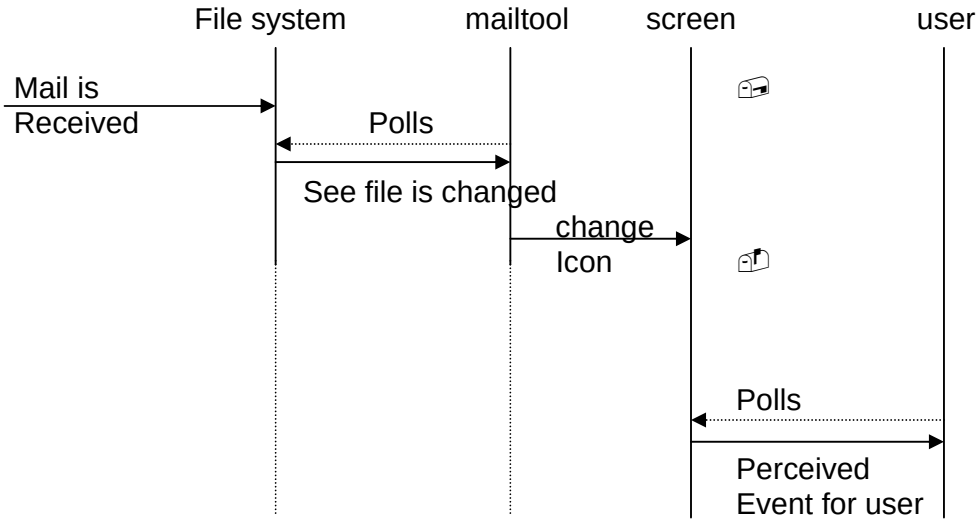
Actual vs perceived : event Brian setelah jam 7:35. Brian tidak melihat jamnya pada saat yang benar. Actual event menjadi perceived event beberapa menit setelah Brian melihat jamnya kembali.

Granularity : jam menunjukkan 7:35 dan ulang tahun Brian merupakan event, tetapi beroperasi pada rentang waktu yang berbeda.

Implikasi pada perancangan
Agar event yang dicapai user dapat pada skala waktu yang tepat, kita harus dapat memprediksi skala waktu event pada teknik interface yang bermacam2. Mempresentasi informasi sederhana pada layer atau menyebabkan event pada interface tidak menjamin bahwa event akan tercapai seperti keinginan user.

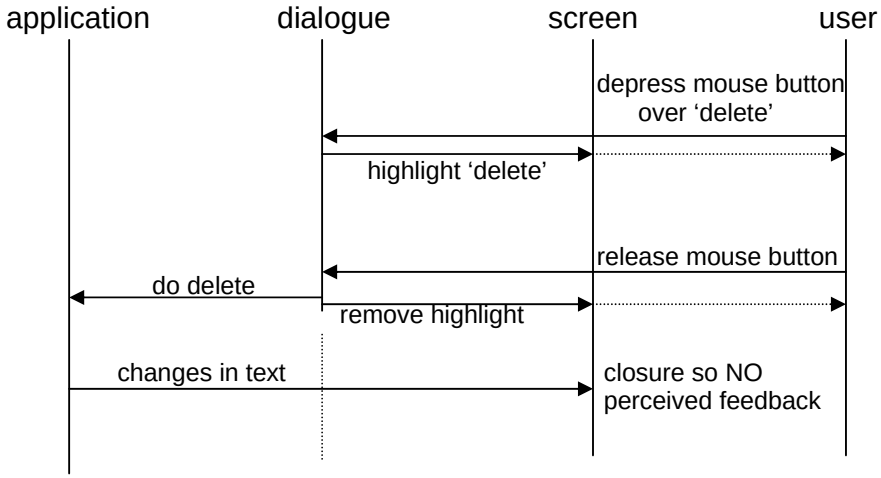
Psikologi naif
Psikologi ini stimuli apa yang penting dan ke mana atensi user difokuskan. Pertama dengan memprediksi ke mana user mencari, yaitu dari menggerakkan mouse, text insertion point, screen. Jika kita mengetahui ke mana user beratensi, kita dapat memberikan informasi. Perubahan focus visual user menjadi penting dan membuat event user tercapai. Kedua adalah event yang segera meskipun kita belum mengetahui ke mana user beratensi. Dan yang terakhir adalah experience closure, merasa telah lengkap melakukan sesuatu dan bersiap untuk sesuatu yang lain. Implikasinya pada persepsi dan aksi.

Contoh : email interface

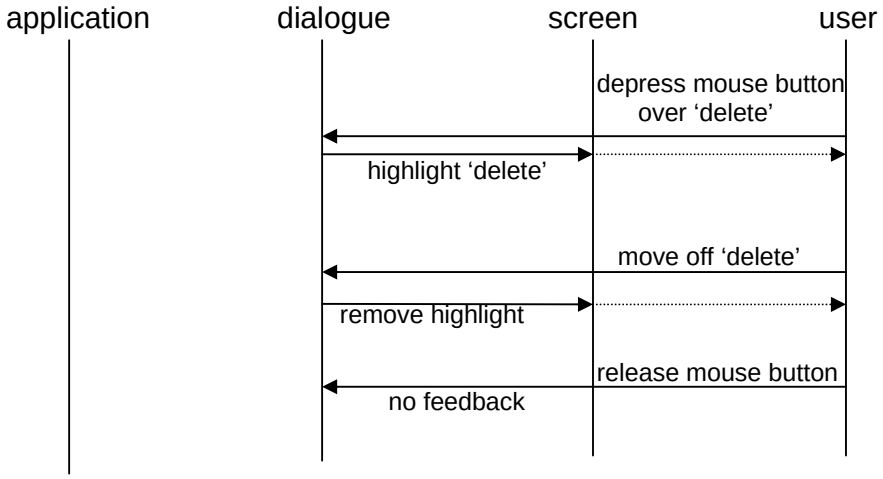


Jika ‘SEND’ dihit, pesan akan terkirim – ini event. Event dari penerimaan mail kemudian direfleksikan ke perubahan status dari system file. Pada workstation yang sedang berada pada mailtool, icon ‘empty mailbox’ terlihat. Mailtool tidak berubah segera, tetapi secara periodic memeriksa file jika adalah yang berubah – disebut polls. Setelah beberapa saat, mailtool poll file dan melihat ada perubahan terjadi. Saat ini mengubah status dari file system menjadi perceived event bagi mailtool. Mailtool perlu memberitahu bahwa mail telah sampai dan harus membuat event ini tercapai bagi user. Mailtool melakukan perubahan pada icon-nya, ini adalah perubahan event ke status. Bagi user, dengan melihat perubahan pada icon, membuat user melakukan polls untuk membuka mail tersebut. Ada beberapa interface lainnya seperti explicit examination, audible bell, moving faces.

Screen button feedback (hit)



Screen button feedback (miss)



Screen button yang diaktifkan dengan mengklik mouse merupakan ‘widget’ standard dalam setiap toolkit interface dan banyak ditemukan pada aplikasi interface yang modern. Masalah yang umum adalah bahwa user berpikir button (tombol) telah ditekan tetapi sebenarnya belum. Misalnya pada pemroses kata. Dengan melakukan delete, mouse harus dipegang dan salah satu tombolnya ditekan dengan jari agar proses delete dapat berlangsung. Event ini langsung menuju ke dialogue yang merespon dengan menunjuk/ menerangi button ‘delete’. Jika penekanan tombol terjadi terhadap icon ‘delete’, user akan menemukan bahwa apa yang tidak diinginkan akan terhapus. Tetapi jika penekanan tombol mouse terhadap icon delete tidak dilakukan (tidak jadi), maka perintah delete menjadi batal dan tidak terjadi respon yang berarti.

Latihan

1. Simbol-simbol dasar yang ada dalam logika temporal, kecuali :

a. ~ c. □
b. + d. ◇

2. Dalam model PIE, model black box untuk display adalah :

a. P → E c. E → R
b. E → D d. P → D

3. Apa yang user berikan tentang efek merupakan interpretasi WYS yang :

a. WYou will Get c. WYou have Got
b. WI See d. Tidak ada yang benar

4. Model-model interaksi, kecuali :

a. Windowing system c. Non determinism
b. Dynamic pointers d. Status/ Event analysis

5. Sesuatu yang selalu memiliki nilai yang dapat direfer disebut sebagai :

a. Status c. Event
b. Polling d. Granularity