MODEL- MODEL SISTEM

1. **Formalisasi standard dari RPL untuk membuat sistem yang interaktif**
2. **Model-model interaksi khusus**
3. **Model interaksi yang lengkap.**

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 mudah 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 : teori - praktek
- **Periksa konsistensi eksternal,**

Yang berhubugan dengan program (bukan keuntungan dalam IMK)
- **Periksa konsistensi eksternal,**

Yang berhubungan dengan kebutuhan2, beberapa diantaranya seperti properti keamanan, sistem khusus, dll.

Notasi Berorientasi Model
Dimulai pada akhir 1970 dan 1980.
Ada 2 yang digunakan : Z dan VDM, digunakan untuk spesifikasi interface.

❖ **Simple Sets**

Set yang paling sederhana (standard) : R, Z, N
Yang non standard adalah 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

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 Shape dapat dideklarasikan sebagai s ⟶ s.wid, s.ht
(pascal dengan record, C dengan struct)

Deretan (sequence) dapat mempunyai panjang tetap seperti array. Dalam matematika panjang deretan bisa bervariasi.
a ⌒ b – tipe list dalam LISP.

Function : perhitungan standard dalam bahasa pemrograman. Fungsi memetakan elemen2nya dari satu himpunan ke himpunan lainnya. Contoh : sqrt atau log
Fungsi untuk grafik,
misal Shape_dict == [id] ⟶ **Shape**

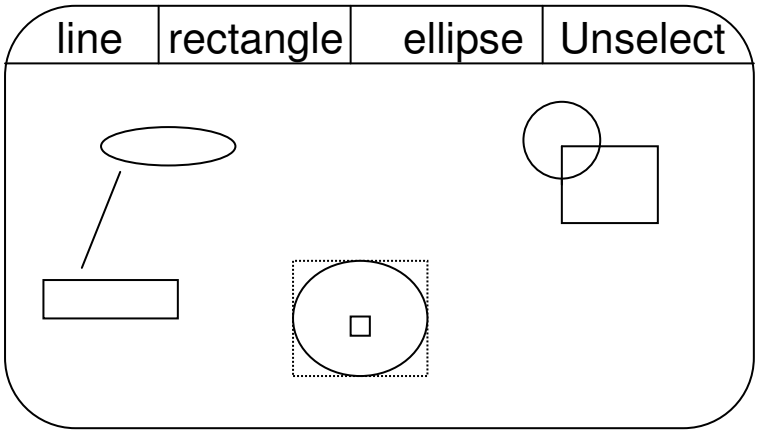
Shape(id).type = rectangle
Shape(id).wid = 2.3
Shape(id).ht = 1.4
Shape(id).centre = (1.2, -3.0)

Shape_dict sebagai fungsi parsial. Fungsi parsial tidak dapat memetakan semua elemen sumber ke elemen himpunan tujuan. Himpunan nilai yang valid : 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**

State dan operator dalam Z ditulis dalam notasi skema.



SKEMA

State	identifikasi/ definisi dari
shapes : shape_dict	komponen sistem grafik
selection : P id	
selection $\subseteq$ dom shapes	invariant – kondisi yang harus dipenuhi komponen state

Init	Sistem model geometri.
State	State tanpa bentuk (shapes)
dom shapes = { }	yang dibuat/ dipilih
selection = { }	

Init	Definisi dari state awal
State	yang menghilangkan predikat
dom shapes = { }	

Mendefinisikan Operasi

Untuk mendefinisikan operasi, perlu dibuat state dari sistem grafik.

State ——— sebelum kopi **? —— input**
State' ——— sesudah kopi **!——— output**

Membuat New Ellipse yang berukuran tetap.

New_Ellipse
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? ⟶ Newshapes?}
selection' = {newid?}

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

♦ **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)

'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.

❖ **Notasi Aljabar**

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.

Algebraic-draw =
types
State, Pt
operations
init : ⟶ state
new_ellipse, new_rectangle,
new_line : Pt x State ⟶ State
move : Pt x State ⟶ State
unselect : State ⟶ State
delete : State ⟶ State
axioms
for all st ∈ State; p, p' ∈ 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)

resize(p, move(p', new_rectangle(st)), berarti ?
axiom 1 dan 5 dapat dituliskan
new_ellipse; delete = unselect
unselect; resize(p) = unselect

❖ **Logika Temporal dan Lainnya**
(p ∧ q) ∨ r , dimana
p = 'my nose is green'
q = 'I've got ears like a donkey'
r = 'I'm called Alan'

disebut logika predikat dan logika proporsional (P(x) ∨ Q(x))

Simbol-simbol pada logika temporal.
Simbol dasar adalah □ ◇ ∀ ∃ ¬ (always, eventually, for all, there exist, not)

□¬ = never

contoh :
□(rains on Tuesday)
□(¬p) it is always not true when p = p never happens
□¬(computer explodes)
□(user types 'print fred' ⇒ ◇ the laser printer prints the file 'fred')

Operator tambahan
Eventually ⟶ before the end of this interval
p until q – p harus benar sampai q benar
p before q – p harus benar pada beberapa saat sebelum q benar

p until q lebih lemah dari □p
p before q lebh kuat dari ◇p.

(◇ user types 'print fred') ⇒ the laser printer prints the file 'fred'

❖ **Logika Deontic**
Konsep agent (human, corporate dan computer) yang bertanggungjawab dan saling ketergantungan di antara agent.
Operator umum : permission (per), obligation(obl)
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')

request('print fred'), dapat dilakukan oleh agent.

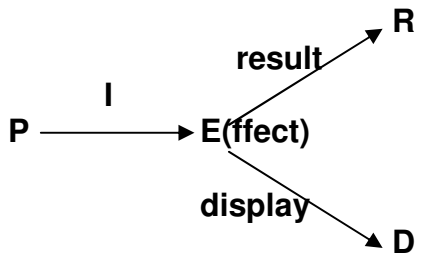
performs(Jane, request('print fred')) ⇒ obl(lp3, print(the file 'fred'))

Model-Model Interaksi

WYSWYG – generik, model formal dari system yang interaktif.

❖ **Model PIE**
Model black box,
Display : E ⟶ D, result : E ⟶ R; fungsi interpretasi, I : P ⟶ E

Aksi user memberi perintah (commands = C) yang biasa disebut Program.



doit : E x P ⟶ E (doit(e,p))
fungsi interpretasi, I,
doit(I(p),q) = I(p⌒q)
doit(doit(e,p),q) = doit(e, p⌒q)

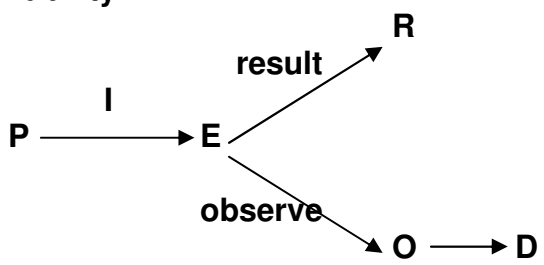
C : keystrokes atau klik mouse
C = {'a', 'b', …, '0', '1', …, '*', '&', …}
D : display fisik
D = pixel_coord ⟶ RGB_value
R : output yang dicetak
R = inkon paper

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**



WYSWYG 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).

Efek Observasi(lihat gambar sebelumnya).

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

Efek observasi memuat minimal informasi sebanyak hasilnya. Juga berisi informasi tambahan dari keadaan interaksi sistem.

Kondisi yang lebih kuat (fully predictable).

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

❖ **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

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

- **Model-model Interaksi Lainnya**
  - **Windowing system**
  - **Timing**
  - **Attention**
  - **Non determinism**
  - **Dynamic pointers**

Model PIE merupakan model event-in/ status-out.

Status / Event Analysis

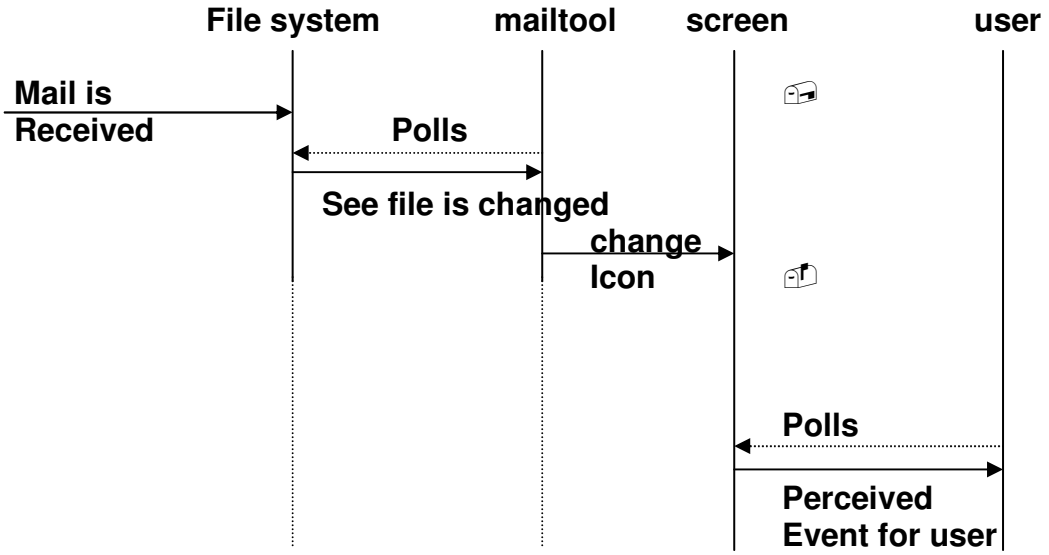
Perbedaan status dan event adalah being dan doing.
Properti event : waktu dan kalender.
Misal : jam digital, hari ulang tahun, …

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.
Contoh : mouse, text insertion point, screen

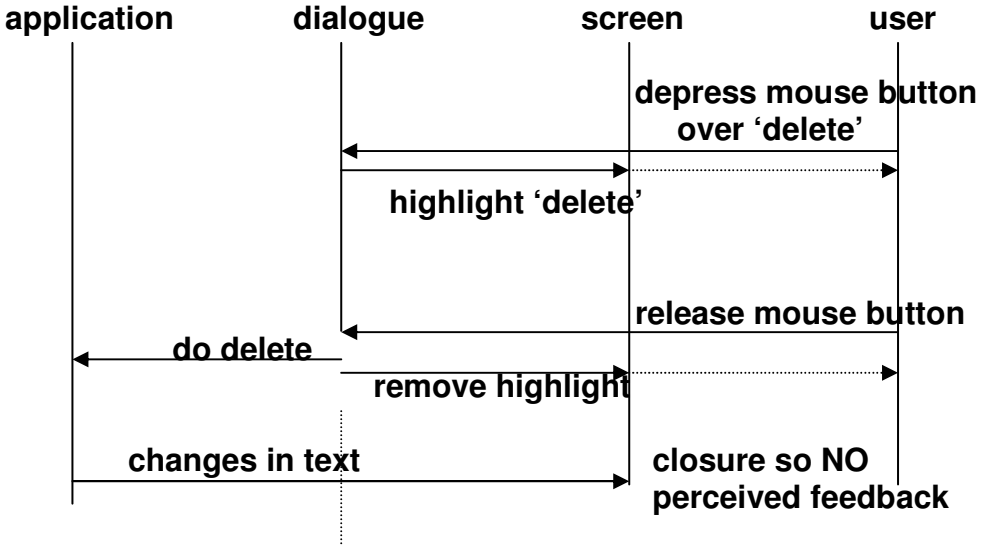
Jika kita mengetahui dimana dan kemana user harus melihat, maka kita dapat memberikan informasi. Perubahan pada fokus visual user penting dan menjadi event yang dicapai user.

Experience closure : merasa telah lengkap melakukan sesuatu dan bersiap untuk sesuatu yang lain. Implikasinya pada persepsi dan aksi.

Contoh : email interface



Screen button feedback (hit)



Screen button feedback (miss)