PERKULIAHAN KE 9

Tujuan Instruksional Khusus (TIK)

- Menjelaskan elemen sistem windowing
- Menjelaskan pemrograman aplikasi dan penggunaan alat bantu
- Menjelaskan sistem manajemen user interface
- Menjelaskan beberapa kebutuhan untuk mendukung user
- Menjelaskan beberapa pendekatan untuk mendukung user
- Menjelaskan apa yang dimaksud dengan sistem bantu pintar serta beberapa model & teknik sistem ini
- Menjelaskan perancangan sistem pendukung user.

Pokok Bahasan : Elemen sistem windowing, Pemrograman Aplikasi, Menggunakan Alat bantu, Sistem Manajemen User Interface, Kebutuhan untuk mendukung user, Pendekatan untuk mendukung user, Sistem bantu pintar (intellegent), Perancangan sistem untuk mendukung user.

Deskripsi Singkat : bahasan ini tentang pendukung pemrograman yang disediakan untuk implementasi dari sebuah sistem yang interaktif. Juga memberikan penjelasan bagi pengguna dalam membuat suatu sistem bantuan-help dan bagaimana membuat dokumentasi dari sistem.

Bahan Bacaan :

Dix, Alan et.al, *HUMAN-COMPUTER INTERACTION*, Prentice Hall, Europe, 1993, hal 337-361, 401-419

Newman, W. M and Lamming, M. G, *Interactive System Design*, Addison Wesley, Cambrigde, Great Britain, 1995

IMPLEMENTATION SUPPORT

Tujuan

- Tool pemrograman untuk system interaktif :
  - Menterjemahkan desain abstrak secara efektif dan prinsip usability ke bentuk yang dapat dieksekusi
  - pelayanan pada level yang berbeda bagi pemrogram

- Sistem window :
  - Lingkungan utama bagi pemrogram dan user
  - Memungkinkan workstation tunggal mendukung system user yang terpisah melalui aksi yang simultan

- Toolkit interaksi :
  - Memungkinkan pemrogram untuk menggambarkan perilaku objek pada level yang sama terhadap pencapaian user

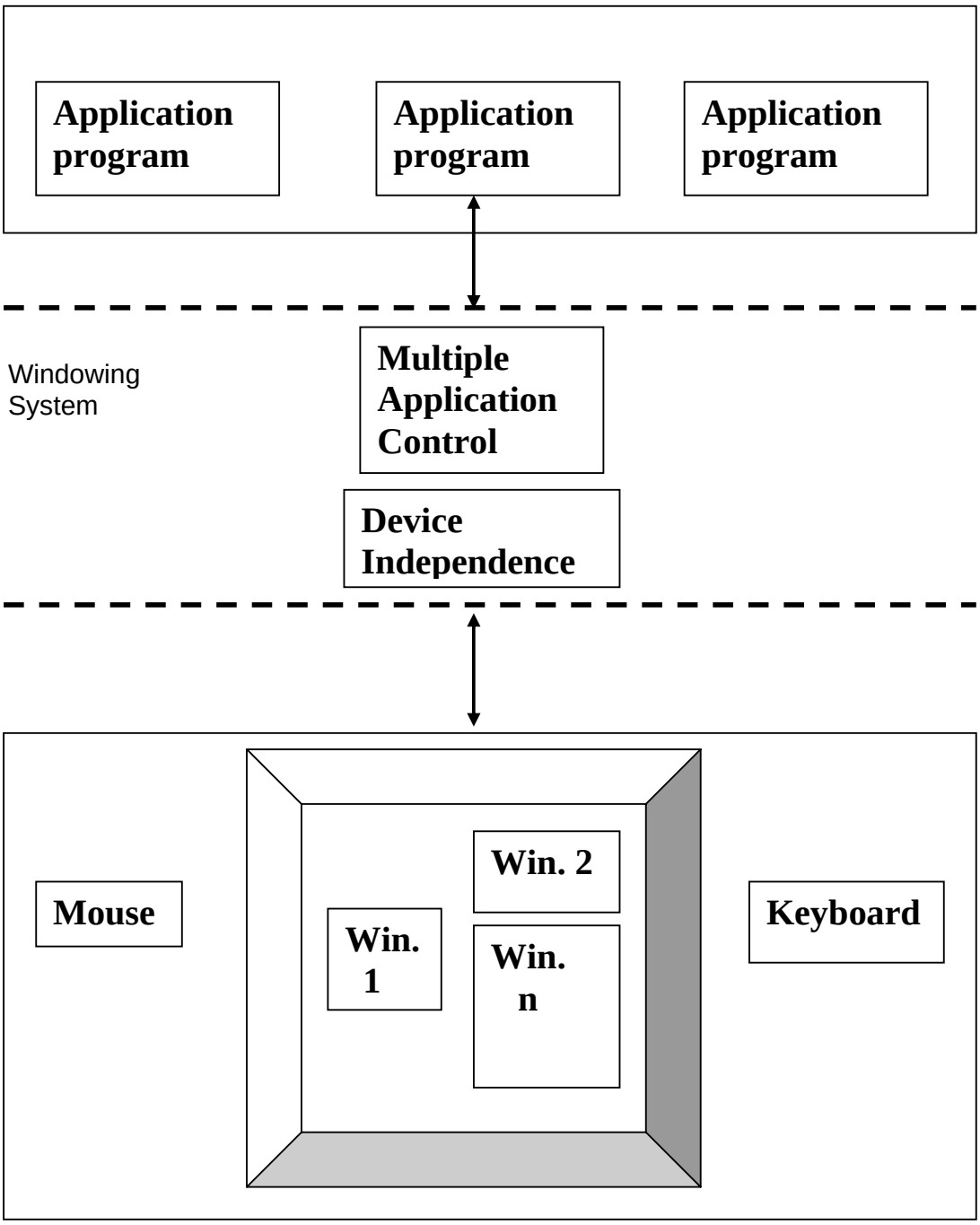
- Sistem Manajemen Interface User/ User Interface Management Systems (UIMS) :
  - Level terakhir dari tool pendukung pemrograman
  - Memungkinkan desainer dan pemrogram untuk mengontrol relasi antara objek presentasi dari toolkit dengan semantic fungsionalnya dalam aplikasi sebenarnya.

Element Sistem Window

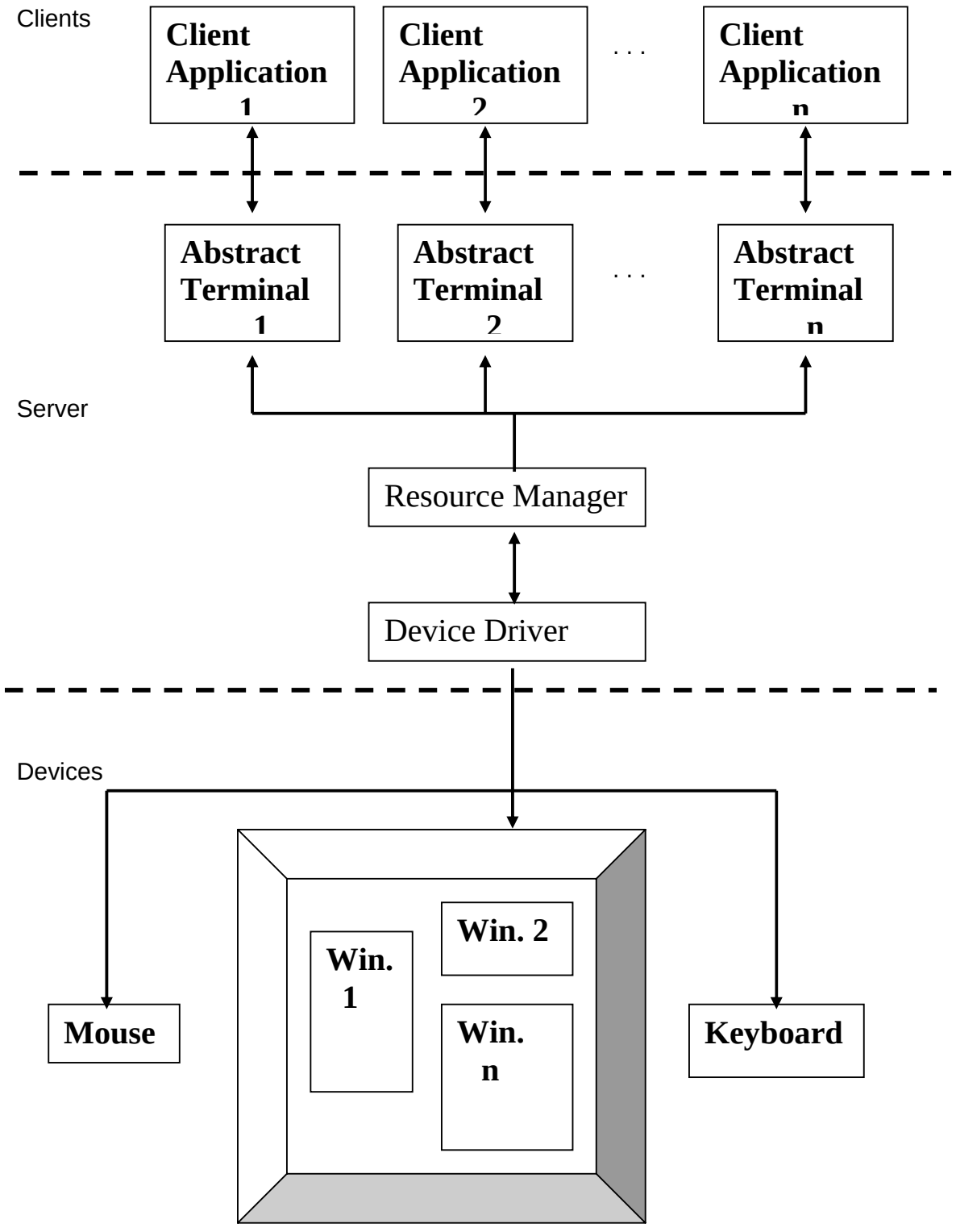
Dua sifat dari system window
Kebebasan dari perangkat keras. Workstation khusus akan berinteraksi dengan beberapa layar display visual, papan ketik dan biasanya beberapa perangkat penunjuk seperti mouse. Keberagaman dari perangkat keras ini dapat digunakan pada setiap system interaktif dan semuanya berbeda dalam hal data yang dikomunikasikan dan perintah yang digunakan. Untuk itu, pemrogram membutuhkan suatu perintah langsung ke suatu terminal abstrak yang mengerti dengan bahasa yang generic dan dapat diterjemahkan ke bahasa dari banyak perangkat khusus lainnya. Selain membuat tugas pemrograman lebih mudah, terminal abstrak memungkinkan portabilitas dari program aplikasi. Hanya satu program terjemahan – device driver – butuh dituliskan untuk perangkat keras khusus dan kemudian setiap program aplikasi dapat mengaksesnya. Bahasa generik untuk terminal abstrak pada system window disebut dengan imaging model, beberapa diantaranya : pixels, graphical kernel system (GKS), programmer's hierarchical interface to graphics (PHIGS), postscript.

Sistem window menyediakan kemampuan berbagi sumber dari satu konfigurasi perangkat keras dengan beberapa salinan terminal abstrak. Masing2 terminal abstrak berlaku sebagai proses bebas dan system window akan mengkoordinasikan control dari proses yang ada. System window juga perlu untuk menampilkan aplikasi yang terpisah dengan mendedikasikan daerah dari layar display ke setiap terminal

abstrak. Tugas koordinasi berhubungan dengan menyelesaikan konflik display ketia daerah layar yang terlihat dari dua terminal abstrak saling tumpang tindih.



Arsitektur dari Sistem Window



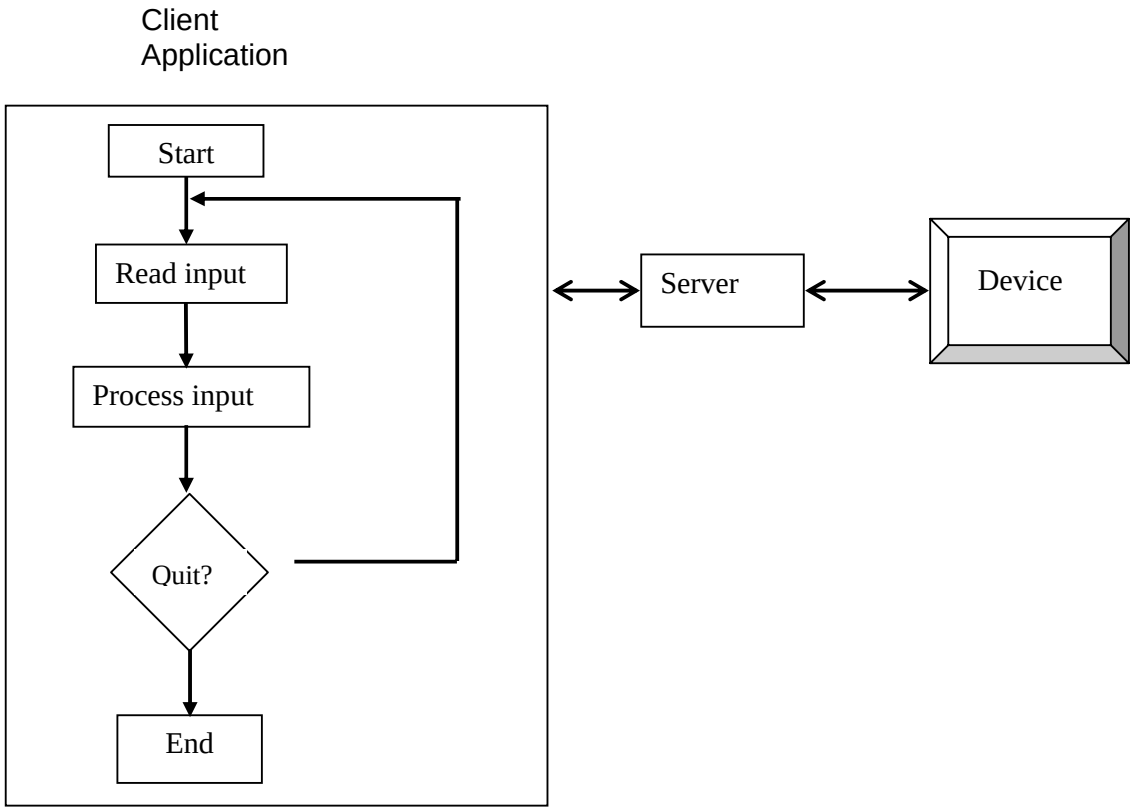
Bass dan Coutaz mengidentifikasikan ada 3 arsitektur yang mungkin bagi perangkat lunak untuk mengimplementasikan role dari system window. Semua ini diasumsikan bahwa device driver terpisah dari program aplikasi. Pilihan pertama adalah untuk

mengimplementasikan dan replikasi manajemen dari proses yang multiple dalam setiap aplikasi yang terpisah. Arsitektur ini tidak terlalu baik karena mendorong setiap aplikasi untuk melihat masalah yang sulit dari penyelesaian konflik sinkronisasi dengan perangkat hardware yang berbagi. Juga mengurangi portabilitas dari aplikasi yang terpisah. Pilihan kedua adalah mengimplementasikan aturan manajemen dalam kernel system operasi, memusatkan tugas manajemen dengan membebaskan dari aplikasi individual. Aplikasi masih harus dibangun dengan system operasi khusus. Pilihan ketiga adalah portabilitas, fungsi manajemen ditulis sebagai aplikasi yang terpisah sehingga dapat menyediakan interface ke program aplikasi lain yang generic terhadap semua system operasi. Pilihan terakhir adalah model arsitektur client-server seperti gambar di atas. Dalam prakteknya, pembagian dari arsitektur ini tidak terlalu jelas dan setiap aplikasi interaktif atau kumpulan operasi aplikasi dalam system window berbagi fitur dengan salah satu dari ketiga aritektur konseptual tersebut. Sehingga, perlu ada satu komponen yaitu aplikasi atau proses yang terpisah bersama dengan beberapa pendukung system operasi yang siap dan pendukung aplikasi yang hand-tuned untuk menangani sumber bersama. Aplikasi yang dibuat untuk system window yang berbasis pada model client-server tidak terlalu portable. Contoh dari system window berbasis arsitektur client-server adalah system window X release 11 standar industri X11, dibuat di MIT pertengahan 1980an.

Memprogram Aplikasi

Aplikasi yang interaktif umumnya user-driven, aksi aplikasi yang ada ditentukan oleh input yang diterima dari user. Ada 2 paradigma pemrograman yang dapat digunakan untuk mengorganisasikan alur control dalam aplikasi. Paradigma pertama adalah Read-Evaluation Loop, yang internal terhadap program aplikasi itu sendiri. Contoh pada pemrograman Macintosh. Server mengirim input user sebagai event terstruktur ke aplikasi client. Fokus server yang penting adalah pada event dari client yang harus diarahkan. Aplikasi client diprogram untuk membaca setiap event yang melaluinya dan menentukan semua perilaku aplikasi khusus yang menghasilkan respon. Alur logika dari aplikasi client dalam pseudocode dan diagram adalah :

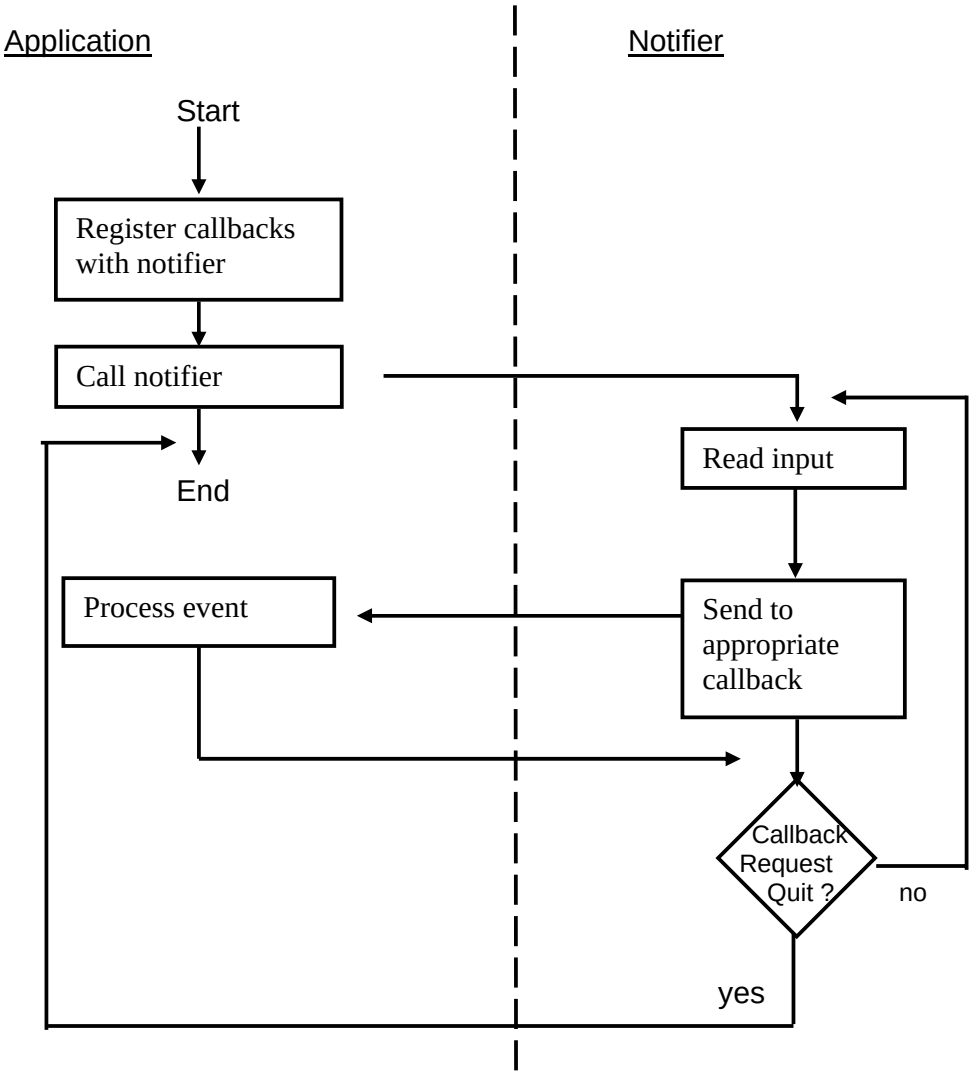
```
repeat
        read-event(myevent)
        case myevent.type
                type_1 :
                        do type_1 processing
                type_2 :
                        do type_2 processing
                        .
                        .
                type_n :
                        do type_n processing
        end case
end repeat
```



Aplikasi memiliki control yang lengkap terhadap proses event yang diterima. Pemrogram harus mengeksekusi control melalui setiap kemungkinan event yang client akan terima. Pada macintosh, MacApp.

Paradigma pemrograman lainnya adalah berbasis notifikasi, dimana loop control utama untuk proses event tidak ada dalam aplikasi. Pusat notifier menerima event dari system window dan menyaringnya ke program aplikasi dengan suatu program seperti terlihat pada gambar di bawah. Program aplikasi menginformasikan ke notifier event apa yang penting dan masing2 event mendeklarasikan satu prosedurnya sebagai callback sebelum mengubah kontrolnya ke notifier. Ketika notifier menerima event dari system window, terlihat jika event diidentifikasi oleh program aplikasi maka notifier akan melewatkan event dan control ke prosedur callback yang diregistrasi untuk event. Setelah pemrosesan, prosedur callback mengembalikan control ke notifier, memberitahukan untuk melanjutkan event yang diterima atau meminta diakhiri.

Alur control terpusat di notifier yang membebaskan program aplikasi dari proses yang terlalu banyak dari setiap proses event yang lewat dari system window. Misalkan program aplikasi akan menghasilkan kotak dialog pre-empsi dan menginginkan adanya konfirmasi dari user sebelum diproses. Dialog pre-empsi menghapus secara efektif semua aksi user kecuali yang dibutuhkan user untuk memperbaikinya.



Pada paradigma loop read-evaluation, ini langsung dikerjakan. Jika kesalahan dideteksi, aplikasi memulai loop read-evaluation yang ada dalam cabang statement case. Pada loop tersebut, semua event yang tidak relevan dapat diterima dan dihapus. Pseudocode nya adalah :

```
repeat
        read-event(myevent)
        case myevent.type
                type_1 :
                        do type_1 processing
                type_2 :
                         if (error-condition) then
                        repeat
                                read-event(myevent2)
                                case myevent2.type
                                        type_1 :
                                                .
                                                .
```

```
                                    type_n :
                                end case
                            until (end-condition2)
                            end if
                            …..
                        type_n :
                            do type_n processing
                end case
        until (end-condition)
```

Pada paradigma berbasis notifikasi, dialog pre-empsi tidak sederhana, karena alur control diluar dari pemrogram aplikasi. Prosedur callback harus dimodifikasi semua untuk megenal situasi dimana dialog pre-empsi diperlukan dan pada situasi tersebut dihapus semua event yang dilewatkan kepadanya oleh notifier.

User Interface Management Systems (UIMS)

Set dari pemrograman dan teknik desain yang dapat menambah level lain dari servis untuk desain system interaktif selain level toolkit adalah system manajemen interface user (UIMS) ini. Focus utama dari UIMS :

- Arsitektur konseptual untuk struktur dari system interaktif yang dikonsentrasikan pada pemisahan semantic aplikasi dan presentasi
- Teknik untuk mengimplementasikan aplikasi dan presentasi secara terpisah
- Teknik pendukung untuk menangani, mengimplementasikan, dan mengevaluasi lingkungan interaksi yang sedang berjalan.

UIMS sebagai arsitektur konseptual
Isu utama adalah bagaimana memisahkan antara semantic aplikasi dan interface yang tersedia bagi user. Banyak argument yang baik untuk mendukung pemisahan ini, yaitu :
Portability : agar aplikasi yang sama dapat digunakan di system yang berbeda maka membuat aplikasinya sebaiknya terpisah dari interface device-dependent-nya.
Reusability : pemisahan meningkatkan komponen untuk dapat digunakan kembali agar dapat mengurangi biaya.
Multiple interfaces : untuk meningkatkan fleksibilitas aplikasi yang interaktif, beberapa interface yang berbeda dibuat untuk mengakses fungsionalitas yang sama.
Customization : interface user dapat dikustom oleh desainer dan user untuk meningkatkan keefektifan tanpa mengubah aplikasi.

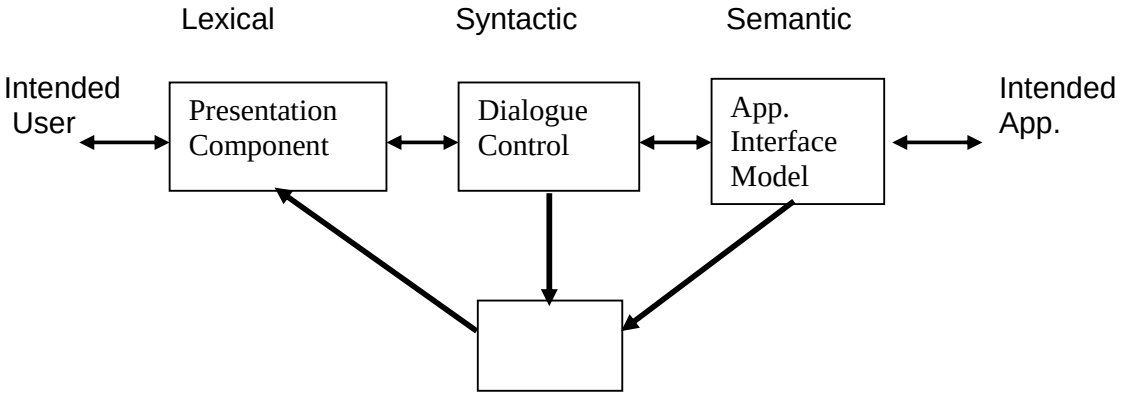
Sekali aplikasi dan presentasi dipisahkan, komunikasi antara keduanya perlu dipertimbangkan, ini yang disebut sebagai control dialog. Secara konseptual, ada 3 komponen utama dari system interaktif – aplikasi, presentasi dan control dialog.

Komponen logika dari UIMS :

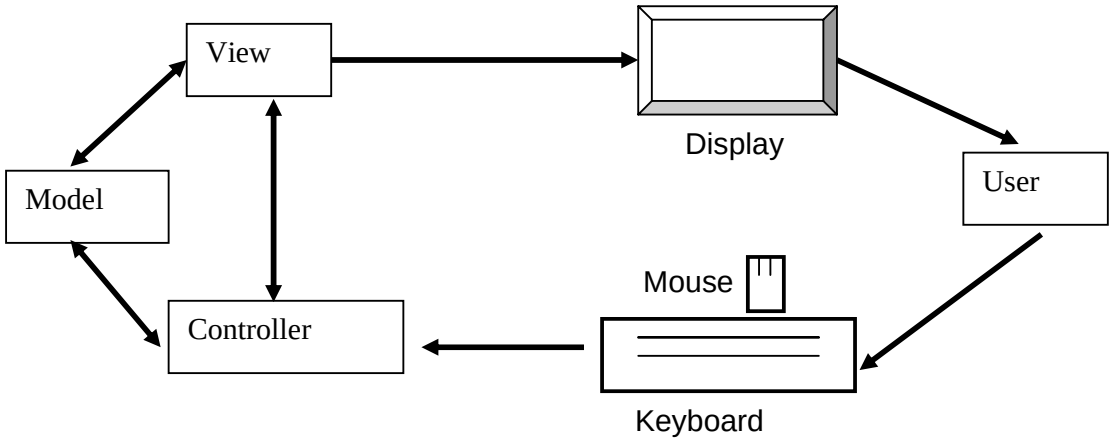
- Presentasi : komponen bertanggungjawab atas tampilan interface, termasuk output dan input yang tersedia bagi user.
- Control dialog : komponen mengatur komunikasi antara presentasi dan aplikasi.

- Interface aplikasi : pandangan dari semantic aplikasi yang disediakan sebagai interface.

Model Seeheim berikut memasukkan aplikasi dan user dalam konteks dari system interaktif meskipun tidak secara eksplisit karena hanya memodelkan komponen logika UIMS bukan system interaktif secara keseluruhan. Dengan tidak membuat aplikasi secara eksplisit ada di model, control dialog eksternal perlu diasumsikan. Dari sudut pandang pemrogram, model Seeheim ini sesuai dengan adanya pembedaan antara lapis leksikal klasik, sintaksis dan semantic dari system computer. Masalah utama dari model Seeheim ini adalah meskipun terlayani baik pada akhirnya, tetapi tidak terlihat bagaimana arah sebenarnya dari kemungkinan UIMS distrukturkan. Gambar tersebut menunjukkan alasan efisiensi yang mungkin dengan dilewatkan/ dihindarkannya komponen control dialog secara eksplisit sehingga aplikasi memberikan respon semantic aplikasi yang lebih besar. Kotak kosong tersebut ada karena logika tidak dipisahkan dari implementasi. Selain itu model Seeheim tidak menginformasikan bagaimana membangun system interaktif yang besar dan kompleks dari komponen yang lebih kecil.

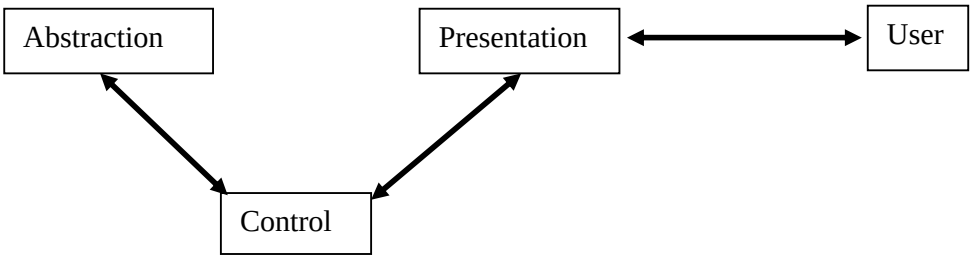


Paradigma Model-View-Controller (MVC) menangani masalah Seeheim di atas. Pada pemrograman Smalltalk, link antara semantic aplikasi dan presentasi dapat dibangun dari tiga serangkai MVC ini. Smalltalk adalah system pemrograman berorientasi objek yang berhasil membangun system interaktif baru berdasarkan system yang sudah ada.



Model MVC menunjukkan semantic aplikasi, view menangani output grafik atau teks dari aplikasi dan pengontrol menangani input. Perilaku dasar dari model ini adalah view dan pengontrol ditempelkan/ dimasukkan dalam kelas objek umum dari Smalltalk yang diwariskan dari instant dan dimodifikasi.

Model Coutaz berikut merupakan system interaktif berasitektur multi-agent, disebut model Presentation-Application-Control (PAC). Model ini berbasis tiga serangkai pula, semantic aplikasi dilambangkan dengan komponen abstraksi, input dan output digabungkan dalam satu komponen presentasi dan ada komponen control eksplisit yang menangani dialog dan menghubungkan aplikasi dan presentasi.



Ada 3 perbedaan penting antara PAC dan MVC. Pertama adalah PAC menggabungkan input dan output, MVC memisahkannya. PAC menyediakan komponen eksplisit yang tugasnya melihat kekonsistenan antara abstraksi dan presentasi, dimana MVC tidak menugaskan ke salah satu komponen. PAC tidak berhubungan dengan lingkungan pemrograman apapun, secara kondusif merupakan pendekatan berorientasi objek. Perbedaan yang terakhir ini yang membuat PAC mudah mengisolasi komponen control; PAC lebih merupakan arsitektur konseptual dibandingkan MVC karena kurang implementation-dependent.

Latihan

1. Yang bukan merupakan komponen dari UIMS adalah :

a. Presentasi
b. Interface aplikasi
c. Kontrol dialog
d. Viewer

2. Yang termasuk elemen dari sistem window, kecuali :

a. Device driver
b. Resource sharing
c. Terminal
d. Toolkit

3. Dalam model Seeheim, level sintaksis merupakan komponen logika UIMS untuk :

a. Input
b. Presentasi
c. Kontrol dialog
d. Interface aplikasi

4. Komponen logika pada UIMS pada presentasi merupakan:

a. Sebuah komponen yang bertanggung jawab pada penampilan dari interface.
b. Sebuah komponen yang mengatur komunikasi antara presentasi dan aplikasi.
c. Pandangan dari sebuah aplikasi semantik yang menyediakan sebuah interface
d. Teknik yang mendukung untuk mengatur, implementasi dan evaluasi sebuah run time interaction environtment.

5. Komponen logika pada UIMS pada kontrol dialog merupakan:

a. Sebuah komponen yang mengatur komunikasi antara presentasi dan aplikasi.
b. Pandangan dari sebuah aplikasi semantik yang menyediakan sebuah interface
c. Teknik yang mendukung untuk mengatur, implementasi dan evaluasi sebuah run time interaction environment
d. Sebuah komponen yang bertanggung jawab pada penampilan dari interface

HELP DAN DOKUMENTASI

Tinjauan

- User mempunyai perbedaan kebutuhan di waktu yang berbeda

- User support seharusnya :
  - Tersedia tetapi tidak mencolok
  - Akurat dan kuat
  - Konsisten dan fleksibel

- Jenis-jenis user support :
  - Command based methods
  - Context-sensitive help
  - Tutorial help
  - On-line documentation
  - Intelligent help

- Merancang user support harus memperhatikan :
  - Presentasi
  - Implementasi

Pendahuluan

Ada sebagian pendapat menyatakan bahwa system yang interaktif dijalankan tanpa membutuhkan bantuan atau training. Hal ini mungkin ideal akan tetapi jauh dari kenyataan. Pendekatan yang lebih membantu adalah dengan mengasumsikan bahwa user akan membutuhkan bantuan pada suatu waktu dan merancang bantuan (help) ini ke dalam system.

- Ada empat jenis bantuan yang dibutuhkan user :
  - Quick reference

  Digunakan sebagai pengingat untuk user dari tool yang detail yang secara dasar sangat familiar dan biasa digunakan. Seperti menggunakan opsi perintah umum, atau mengingatkan user akan sintaks dari perintah.

  - Task-specific help

  Membantu user menghadapi masalah atau tidak pasti dalam mengambil tindakan memecahkan masalah yang khusus atau tidak pasti dalam mengaplikasikan tool.

  - Full explanation

  Suatu alat bantu atau perintah yang dapat membantu memahami secara lengkap. Penjelasan ini mencakup informasi dimana user tidak membutuhkannya pada saat itu.

  - Tutorial

    Khusus untuk user baru yang menyediakan perintah secara step by step bagaimana menggunakan tool.

Setiap tipe pendukung user ini dibutuhkan oleh user pada saat yang berbeda berdasarkan pengalaman user dengan system dan memenuhi kebutuhan yang berbeda. Ada banyak informasi yang user inginkan – definisi, contoh, kesalahan yang dikenal dan informasi memperbaiki kesalahan, opsi perintah dan lain-lain. Beberapa diantaranya ada yang tersedia dalam perancangan interface nya sendiri dan ada yang dimasukkan dalam ‘bantuan’ atau system pendukung.

Perbedaan utama antara system ‘bantuan’ dan dokumentasi adalah bahwa system ‘bantuan’ berorientasi terhadap masalah dan khusus, sedangkan dokumentasi berorientasi terhadap system dan umum.

Kebutuhan Pendukung User

Availability

User dapat menggunakan bantuan pada setiap waktu selama berinteraksi dengan system. User tidak perlu keluar dari aplikasi selama bekerja untuk membuka aplikasi bantuan. Idealnya, bantuan berjalan konkuren dengan setiap aplikasi.

Accuracy dan completeness

Bantuan ini seharusnya menyediakan keakuratan dan kelengkapan system bantuan. Agar refleksi akurasi tersedia pada keadaan system bantuan perlu mencakup keseluruhan system. Kelengkapan sangat penting jika bantuan tersedia untuk digunakan secara efektif. Perancang tidak dapat memprediksi bagian system user yang mana yang membutuhkan bantuan.

Consistency

Seperti diketahui bahwa user membutuhkan tipe yang berbeda dari bantuan untuk digunakan pada tujuan yang berbeda. Hal ini dapat secara tidak langsung menyebabkan system bantuan tidak dapat bekerja. Sistem bantuan yang tersedia harus konsisten terhadap semua system yang ada dan juga pada system itu sendiri. Bantuan online juga harus konsisten dengan dokumentasi kertasnya, konsisten dalam isi, terminology dan bentuk presentasi. Pendukung user internal dalam aplikasi juga perlu konsisten terhadap system.

Robustness

Sistem bantuan ini biasanya digunakan oleh orang yang sedang dalam kesulitan karena system mempunyai perilaku yang tidak diharapkan atau mempunyai kesalahan. Hal ini sangat penting dimana system bantuan seharusnya robust-kuat baik dalam hal memperbaiki kesalahan dan perilaku yang tidak diharapkan.

Flexibility

System bantuan yang fleksibel akan membuat setiap user dapat beinteraksi dalam mencari sesuatu yang sesuai dibutuhkannya. System bantuan yang fleksibel membantu setiap user berinteraksi sesuai dengan keinginannya. Mulai dari perancangan system bantuan yang interaktif secara modular melalui bantuan context-sensitive untuk membuat system bantuan yang dapat diadaptasi atau pintar yang menginfer keahlian dan tugas user.

Unobtrusiveness

System ini seharusnya tidak mencegah user dalam melanjutkan pekerjaannya atau terpengaruh dengan aplikasi user. Pada satu saat, system bantuan teks pada interface yang bukan window dapat menginterupsi pekerjaan user. Untuk menghindari ini digunakan presentasi pada layar yang terpisah. Pada saat yang lain, system bantuan pintar dalam menyediakan bantuan atas inisiatif sendiri, bukan permintaan user.

Pendekatan-pendekatan Pendukung User

Command assistance

Mungkin pendekatan yang umum untuk user support adalah menyediakan bantuan pada level command, user yang membutuhkan bantuan pada command yang khusus dan ditampilkan pada layar bantuan atau pada manual page yang menjelaskan tentang command tersebut.

Contoh pada UNIX *man* help dan DOS *help* command.
Tipe ini sederhana dan efisien jika user mengetahui apa yang user ingin ketahui dalam mencari informasi yang lebih detail. Diasumsikan bahwa user mengetahui apa yang dicari. Pada system computer yang kompleks, ada beberapa perintah yang user ketahui dengan baik dan dapat menggunakannya dan ada pula beberapa perintah yang jarang digunakan.

Command prompts

Menyediakan bantuan ketika user menemukan kesalahan, biasanya dalam bentuk prompt perbaikan. Prompt sangat berguna jika kesalahan sederhana misalnya kesalahan sintaks, tetapi knowledge-pengetahuan dari perintah harus diasumsikan. Bentuk yang lain dari command prompt adalah penggunaan menu dan icon yang dipilih. Ini memerlukan bantuan ke memori seperti membuat secara eksplisit perintah2 apa yang tersedia pada waktu tertentu. Ini diasumsikan juga memerlukan sejumlah pengetahuan tentang kegunaan perintah sehingga pendukung tambahan masih diperlukan.

Context-sensitive help

Berbentuk menu based system yang menyediakan bantuan pada menu option. Mulai dari yang memiliki pengetahuan khusus dari user khusus hingga tersedianya kunci bantuan sederhana yang diinterpretasikan sesuai dengan konteks yang akan dipanggil dan akan ditampilkan. Contoh perintah help pada editor spy dan bantuan Ballons pada Macintosh.

On-line tutorial

Mengijinkan user bekerja melalui aplikasi dasar dalam lingkungan percobaan. User dapat melihat kemajuan sesuai dengan kecepatan dan dapat mengulangi bagian dari tutorial yang diinginkan. User juga dapat merasakan bagaimana aplikasi bekerja dengan bereksperimen langsung dengan contoh-contoh. Kebanyakan on-line tutorial tidak pintar karena tidak mempunyai pengetahuan tentang user dan pengalaman user sebelumnya ataupun domain atau bentuk pembelajaran. On-line tutorial tidak fleksibel dan sering dilupakan, beberapa akan

mengalami kesalahan dalam memperbaiki jawaban masalah, sederhana karena tidak diformat untuk itu.

On-line documentation

Membuat efektif dengan membuat dokumentasi di kertas tersedia di komputer. Ini membuat materi tersedia terus menerus bersamaan dengan pada saat user bekerja bahkan sejumlah besar user bekerja secara konkuren. Dokumentasi dibuat untuk menyediakan deskripsi dari fungsionalitas dan perilaku system secara sistematik.

Sistem Bantuan Pintar - Intelligent help system

Pada system computer yang kompleks atau besar, user akan terbiasa dengan subset dari fungsionalitas, demonstrasi keahlian dalam beberapa aplikasi dan tidak adanya keahlian dengan yang lain, bahkan kesadaran keberadaannya. User yang berbeda-beda akan memiliki kebutuhan dan level pemahaman yang berbeda. System bantuan pintar dibuat untuk mengatasi masalah ini dengan mengadaptasikan bantuan yang disediakan untuk user individual yang membuat permintaan dan secara aktif memberikan pelajaran alternative dari aksi yang user tidak sadari.

Bantuan pintar adalah kasus khusus dari kelas system interaktif yang umum, dikenal dengan system pintar. Bantuan ini meliputi system pintar dengan domain khusus, system tutor yang pintar dan interface adaptif umum.

Dioperasikan dengan memonitoring aktivitas user dan mengkonstruksikan model sesuai dengan user. Model ini termasuk pengalaman, preferences, kesalahan user atau kombinasi dari semuanya. Dengan menggunakan knowledge-pengetahuan ini bersama dengan pengetahuan dari domain dimana user bekerja dan kadang-kadang strategi tutorial atau advisor umum, system bantuan pintar akan mempresentasikan bantuan yang relevan dengan tugas user dan disesuaikan dengan pengalamannya.

Knowledge representation : user modelling

Pada banyak system, model ini adalah pandangan perancang untuk user dan diimplisitkan dalam perancangan. Perancang memiliki pemikiran user ‘khusus’ dan membangun interfacenya. Jika perancang selesai dengan pekerjaan ini, model menjadi sangat efektif. Tetapi perlu diasumsikan bahwa semua user adalah sama dan memiliki permintaan yang sama. Sistem yang lain mengijinkan user untuk membuat model dirinya dimana system dikonfigurasikan. Contoh sederhana pada .profile nya unix yang dapat dieksekusi ketika user masuk ke system dan menseting system dan variabel2 nya sesuai dengan keinginan user, disebut system yang adaptable. Pendekatan yang lain dalam menyediakan system dengan model user dan menggunakannya pada system bantuan pintar, adalah dengan memiliki konstruksi system dan menangani model user berdasarkan data yang dikumpulkan sedikit demi sedikit dari pemantauan interaksi user. Ada sejumlah pendekatan model user dikonstruksi dan dipantau, yaitu :

Quantification

Model yang paling sederhana dari user modelling yang menggunakan jumlah tingkatan dari keahlian yang akan merespon dalam cara yang berbeda. User

ditempatkan pada satu level dan berpindah diantara level lainnya berdasarkan pengukuran kuantifikasi dari keahliannya pada waktu tersebut. Kegiatan2 yang berbeda dibobotkan dan user dinilai berdasarkan bobot dari aktifitas yang diambil. Jika nilai melebihi batas tertentu, user dipindahkan ke level keahlian yang lain dan system beradaptasi untuk itu. Contoh : Mason mengadaptasikan presentasi prompt perintah dari level keahlian user.

Move from Level 1 to Level 2
If
The system has been used more than twice (0.25)
Commands x and y have been used effectively (0.20)
Help has not been accessed this session (0.25)
The system has been used in the last 5 days

Stereotypes

System mengelompokkan user sebagai anggota dari kategori user atau stereotype, yang berbasiskan pada karakteristik user dan kemungkinan sederhana seperti membuat perbedaan antara user baru dan user ahli. Atau yang lebih kompleks seperti membuat stereotype yang berbasiskan pada lebih dari satu informasi.

Ada beberapa cara membuat stereotype. Salah satunya adalah menggunakan informasi seperti perintah yang digunakan dan kesalahan untuk menggolongkan tipe user yang berbeda, kemudian menggunakan aturan untuk mengidentifikasikan stereotype dimana user berada. Pendekatan alternative adalah menggunakan pendekatan mesin pembelajaran seperti jaringan saraf untuk mempelajari contoh dari perilaku user yang bermacam2 dan menggolongkan user berdasarkan kedekatannya dalam mempelajari pelajaran sebelumnya.

Overlay models

Merupakan model yang ideal yang membandingkan perilaku user. Hasilnya ditampilkan dalam dua model atau perbedaan. Keuntungan dari model ini dapat melihat secara pasti bagian dari aktivitas suatu system. Tidak hanya system dapat menyadari apa yang user lakukan, tetapi juga memiliki representasi perilaku yang optimal. Ini menyediakan suatu benchmark dalam mengukur unjuk kerja user, dan jika user tidak mengambil aksi yang optimal ada indikasi pada tipe help atau petunjuk yang diperlukan.

Pendekatan yang serupa adalah digunakan pada error based model dimana system menyimpan rekaman kesalahan dan perilaku sebenarnya dari user serta membandingkannya. Jika perilaku sesuai dengan kesalahan pada catalog, aksi yang sama dapat dilakukan.

Knowledge representation : domain and task modelling

Pendekatan yang umum dari masalah ini adalah mewakili tugas user berdasarkan urutan perintah yang tersedia untuk mengeksekusinya. Sebagaimana pada tugas user, perintah digunakan untuk membandingkan urutan tugas yang disimpan dan

mencocokkan dengan urutan tepat. Jika urutan perintah user tidak cocok maka dibutuhkan bantuan. Pendekatan ini digunakan pada system PRIAM.

Knowledge representation : modelling advisory strategy

Sistem ini kadang mencakup intelligent help yang membuat modelling advisory atau strategi tutorial. Dengan menyediakan system bantuan dengan tipe informasi ini memungkinkan tidak hanya memilik advis yang sesuai untuk user tetapi juga menggunakan metode pemberian advis.

Manusia membutuhkan tipe bantuan yang berbeda bergantung pada pengetahuan dan kondisi. Ini meliputi pengingat-reminder, bantuan dengan tugas khusus-task specific help dan bantuan tutorial. Ada bukti mengindikasikan bahwa keahlian user mengikuti strategi yang berbeda ketika memberi advis ke sesamanya. Ini mencakup menginfer keinginan manusia dalam mencari bantuan dan nasehat pada level tersebut atau menyiapkan sejumlah pemecahan terhadap masalah. Alternatifnya, menempatkan masalah dalam konteks dan menyiapkan 'contoh solusi' berdasarkan konteks tersebut.

Teknik untuk representasi knowledge

Terdapat empat group utama dari teknik yang digunakan dalam knowledge representation untuk intelligent help system :

Rule based techniques

Pengetahuan digunakan untuk mewakili sekumpulan aturan dan kenyataan, yang diimpretasikan menggunakan beberapa mekanisme inferensi. Logika predikat menyediakan mekanisme untuk merepresentasikan informasi deklaratif dan aturan produksi merepresentasikan informasi procedural. Teknik ini digunakan untuk domain yang relatif besar dan dapat mewakili kegiatan yang menampilkan pengetahun.

Contoh:

```
IF
        Command is EDIT file1
AND
        Last command is COMPILE file1
THEN
        Task is DEBUG
        action is describe automatic debugger
```

Frame based technique

Digunakan untuk mewakili situasi yang umum terjadi dan pengetahuan 'default'. Frame merupakan suatu struktur yang berisi slot yang diberi label yang mewakili cirri yang berhubungan. Setiap slot diberi nilai atau nilai default. Input user disesuaikan dengan nilai frame dan jika sesuai menyebabkan beberapa aksi diambil.

Contoh :

```
User
        Expertise level : novice
        Command : EDIT file1
        Last command : COMPILE FILE1
```

Errors this session : 6
Action : describe automatic debugger

Network based techniques

Mewakili pengetahuan tentang user dan system yang merupakan hubungan antara kenyataan. Contoh yang paling umum adalah semantic network. Network merupakan suatu hirarki dan child dapat berhubungan dengan parent-nya. Network dapat digunakan untuk menghubungkan representasi berbasis frame.
Contoh compile yang dapat diperluas dengan semantic network :

CC is an instance of COMPILE
COMPILE is a command
COMPILE is related to DEBUG
COMPILE is related to EDIT
Automatic debugger facilitates DEBUG

Example based technique

Mewakili pengetahuan yang secara implicit ada dalam struktur keputusan dari suatu klasifikasi system. Ini dapat berupa pohon keputusan pada pendekatan pembelajaran induktif atau link dari network pada jaringan saraf. Struktur keputusan dibuat secara otomatis berdasarkan contoh yang dipresentasikan pengklasifikasi. Pengklasifikasi secara efektif mendeteksi cirri saat itu dalam contoh dan dapat menggunakannya untuk mengklasifikasikan input lainnya.
Contoh :

EDIT file1
COMPILE file1

Ini dipelajari sebagai contoh dari tugas khusus yaitu DEBUG.

Masalah dengan knowledge representation dan modelling

Merepresentasikan pengetahuan merupakan issue dalam intelligent help system tetapi tidak tanpa masalahnya. Pengetahuan kadang sulit didapatkan, terutama jika ahli domain tidak ada. Ini akan sulit untuk memastikan kelengkapan dan kebenaran dari pengetahuan berdasarkan kondisi. Bahkan jika pengetahuan tersedia, jumlah pengetahuan yang dibutuhkan menjadi penting, membuat bantuan pintar menjadi opsi yang mahal.

Masalah lain adalah mengintepretasikan informasi yang cocok. Meskipun basis pengetahuan dapat disediakan dengan pengetahuan rinci dari konteks yang diharapkan dan domainnya, selama interaksi informasi yang tersedia hanya log system dari aksi user. Data ini tidak berubah dan berisi pola kegiatan saat itu yang dapat digunakan untuk menginfer urutan tugas.

Masalah lain

Inisiatif

Haruskah user mempertahankan pengawasan yang lengkap terhadap system,
Haruskah system langsung berinteraksi atau
Haruskan penggabungan dialog didukung ?

Effect

Para perancang seharusnya memperhatikan efek dari modelling dan adaptasi

Scope

Para perancang perlu memperhatikan scope dari bantuan dimana digunakan pada level aplikasi atau system yang luas.

Merancang Sistem Pendukung User

Ada banyak cara untuk merancangnya dan semua itu diserahkan pada perancang untuk memilih cara yang terbaik akan tetapi hal yang perlu diperhatikan adalah :

- Perancangannya seharusnya tidak seperti “add-on” ke perancangan system. Secara ideal seharusnya merupakan bagian integral dalam sistem
- Perancang harus memperhatikan isi dari bantuan dan konteks yang akan digunakan sebelum teknologi disiapkan.

Masalah presentasi

- *How is help requested ?*
  Pilihan pertama bagi perancang untuk membuat bagaiman bantuan dapat diakses oleh user. Terdapat beberapa pilihan. Bantuan ini dapat berupa command, tombol fungsi yang dapat memilih on atau off atau aplikasi yang terpisah.

- *How is help displayed ?*
  Bagaimana bantuan akan dapat dilihat oleh user. Dalam system window akan ditampilkan dalam window yang baru. Dalam system lain mungkin dalam layar yang penuh atau bagian dari layar. Alternatif lain dapat berbentuk pop-up box atau tingkat command line. Tipe presentasi yang sesuai bergantung pada besarnya tingkat bantuan yang diberikan dan ruang yang dibutuhkan. Membuka halaman manual halaman demi halaman tidak membantu. Beberapa system bantuan aktif menyediakan petunjuk visual ketika memiliki saran pembuatan, ini yang membuat user mengambil saran tanpa meninggalkan atau menginterupsi pekerjaannya.

- *Effective presentation of help*
  Layar bantuan dan dokumentasi seharusnya dibuat sejalan dengan interface yang dibuat, memasukkan kemampuan dan kebutuhan tugas user. Tidak menjadi masalah teknologi apa yang digunakan untuk membuatnya akan tetapi yang perlu diperhatikan dan menjadi suatu prinsip yakni bagaimana menulis dan menampilkan secara efektif. Bantuan dan tutorial ditulis secara jelas, dalam bahasa yang dimengerti, menghindari logat khusus. Jika manual kertas dan tutorial ada, terminology harus konsisten dengan materi pendukung yang online. Materi instruksional membutuhkan bahasa instruksional dan system bantuan memberitahu user bagaimana menggunakan system, tidak hanya menggambarkan system.

Masalah implementasi

Para perancang harus membuat keputusan untuk implementasi berupa hambatan/ batas fisik maupun pilihan yang tersedia untuk user. Keputusan ini sudah termasuk dalam pertanyaan : akankah bantuan merupakan perintah system operasi, apakah berbentuk meta-command atau aplikasi. Hambatan fisik apa yang membuat mesin menentukan screen space, kapasitas memori dan kecepatan. Kecepatan adalah hal

yang penting karena waktu respon yang lambat untuk membuat system tidak digunakan meskipun sangat baik dirancang. Akan lebih baik menyediakan fasilitas bantuan sederhana yang merespon dengan cepat dibandingkan yang canggih yang membutuhkan waktu dalam memberikan solusi.

Masalah lain adalah bagaimana struktur data bantuan : apakah berbentuk single file, hierarchy file atau database. Ini bergantung pada tipe dari bantuan tersebut dibuat, tetapi setiap struktur harus fleksibel dan dapat diperluas – system tidak statis dan topic baru tidak terhindarkan untuk ditambahkan ke system bantuan. Struktur data yang digunakan akan menentukan dan menyediakan tipe pencarian atau strategi navigasi.

Perancang harus mempertimbangkan bahwa pengarang materi bantuan harus sebaik usernya. Bahkan jika perancang menulis teks bantuan awal, ini akan diperluas oleh pengarang lainnya pada waktu yang berbeda.

Latihan

1. Jenis bantuan (HELP) yang khusus menyediakan perintah secara step-by-step terhadap user baru adalah :

a. Full explanation c. Quick reference
b. Task-specific help d. Tutorial

2. Model user yang menggunakan jumlah tingkatan keahlian adalah :

a. Command prompt c. Command assistance
b. Online documentation d. Online tutorial

3. Yang merupakan jenis-jenis user support, kecuali :

a. Intellegent help c. Command based methods
b. Online documentation d. Quick reference

4. Ada empat jenis bantuan yang dibutuhkan user. Yang termasuk Quick reference digunakan sebagai:

a. Membantu user menghadapi masalah atau tidak pasti mengambil tindakan dalam memecahkan masalah yang khusus.
b. Untuk user baru yang menyediakan perintah secara step by step
c. Pengingat untuk user dari suatu yang detail yang secara dasar sangat familiar dan biasa digunakan.
d. Alat bantu atau perintah yang dapat membantu memahami secara lengkap.

5. Ada empat jenis bantuan yang dibutuhkan user. Yang termasuk tutorial digunakan sebagai:

a. Membantu user menghadapi masalah atau tidak pasti mengambil tindakan dalam memecahkan masalah yang khusus.
b. Untuk user baru yang menyediakan perintah secara step by step
c. Pengingat untuk user dari suatu yang detail yang secara dasar sangat familiar dan biasa digunakan.
d. Alat bantu atau perintah yang dapat membantu memahami secara lengkap