HCI

Interakcija covek racunar ili poznatije samo kao HCI je disciplina koja se odnosi na projektovanje, evalvaciju i implementaciju interaktivnih kompjuterskih sistema koje koriste ljudi pri cemu se proucavaju. Pored prethodno nabrojanih stvari, HCI takodje proucava i stvari kao sto su struktura komunikacije covek-racunar, covekove mogucnosti da koristi taj racunar, performanse zadataka koje zajednicki obavljaju ljudi i racunari itd.

Korisnicki interfejs omogucava korisniku da interaktuje sa programom i pritom ga uposljava kako bi zavrsio neki svoj zadatak. On treba biti tako napravljen da prikvira struktura samog programa i korisnicima prikazuje skup elemenata (na primer widgeta) kao predstavnika procesa resavanja zadataka. Moze se reci da korisnicki interfejs omogucava korisniku da interaktuje sa svojim zadacima.

Na slici iznad su predstavljena dva interfona, jedan stari i jedan novi. Reprezentuju dva razlicita interfejsa iste namene. Cesto se postavlja pitanje sta je bolje? Misljenja su podeljena. Krenucemo od onog levo. Zbog cega je neophodan ekran na novijem interfonu? Pre svega zbog feedback-a. Zbog razlicitosti dugmadi na razlicitim novim interfonima, cesto se desava da prilikom jaceg pritiska dodje do unosenja duple vrednosti. Takodje, moze se desiti da neko dugme bas zahteva jaci pritisak, gde prilikom onog slabog nece doci do validnog unosa. Ako ne bi postojala mogucnost da vidimo sta smo uneli, greske bi bile ceste. Postoje i neki drugi oblici feedback-a, kao sto je recimo zvuk, ali se oni trebaju uzeti sa rezervom. Samo se treba postaviti situacija da se zgrada nalazi u nekom bucnijem delu grada gde se uopste zvuk (feedback) ne bi ni cuo. Naravno, zgrada se moze nalaziti i u "mirnom" delu grada, no opet moze doci do problema. Problem se moze javiti, na primer, kada neki bucniji automobil prolazi u trenutku koriscenja interfona. Nedostatak novog interfona se ogleda i u tome sto korisnik mora na spisku ispod da nadje zeljenu osobu, pogleda njenu "sifru", zapamti je, a potom se vrati na interfejs interfona i unose je. Sve to zahteva dosta vremena. Medjutim, postoje i mnoge prednosti. Neke od prednosti su estetika, dostupnost servisa, mogucnost ispavke pogresno unesenog broja, isplatljivost kod vecih zgrada itd. Takodje, jedna od glavnih prednosti koja gura nove interfone se ogleda u postojanju koda koji se moze uneti, kada se zaboravi kljuc, i lakog ulaska u zgradu. Sto se tice prednosti starog interfona, one se pre svega baziraju na jednostavnosti i brzini koriscenja. Mnogo je lakse kliknuti jedno dugme pored zeljenog imena, nego izvrsiti citavu opseznu proceduru koju zahtevaju noviji interfoni. U isti mah, stari interfoni upravo zbog te svoje jednostavnosti ne zahtevaju postojanje ekrana na koje se korisnici novih

moraju fokusirati. Prednosti novih ujedno predstavljaju i nedostatke starih interfona.

Prethodno napisano predstavlja lepo poredjenje dva moguca resenja, medjutim, cesto ljudi zahtevaju poredjenje pomocu brojeva, odnosno nekih kvantitativnih pokazatelja, jer svi razumeju brojeve i svi razumeju taj odnos manje-vece. Za dobijanje tih kvantitativnih pokazatelja se koriste razlicite prediktivne teorije, odnosno prediktivni modeli, koji nam omogucavaju da na osnovu dobijenih numerickih vrednosti lakse donesemo odluku. Kao jedan od najboljih modela se izdvojio Keystroke Level Modeling, poznatiji kao KLM, pomocu kojeg se definise efikasnost proizvoda, odnosno vreme koje je potrebno za izvrsavanje neke akcije. Sam model se bazira na key-ing-u, pritiskanju tastera, motorici coveka. Daje odgovor na pitanje: Koliko ce coveku trebati da nesto ispritiska, iskuca, tacnije koliko ce coveku trebati vremena da pomocu datog proizvoda izvrsi neku akciju u sekundama? Model se bazira na 5 operatora, tacnije na 5 akcija korisnika koje trebaju da se prepoznaju kod koriscenja interfejsa i cijom sumom se dobija efikasnost koriscenja datog proizvoda za datu akciju.

Kada se zeli videti efikasnost proizvoda za neku akciju, treba se definisati sekvenca dogadjaja, podakcija, koje korisnik vrsi nad interfejsom prilikom izvrsavanja date akcije. Nakon sto se definise data sekvenca, svaka od podakcija se zamenljuje odgovarajucim operatorom iz skupa KLM operatora. Nakon sto se i to odradi, jednostavno dolazi do zamene operatora iz sekvence vremenskim intervalima koji im odgovaraju. Sabiranjem tih vremenskih intervala dobija se efikasnost proizvoda za datu akciju u sekundama.

Operatori su sledeci:

- Keying - koliko korisniku treba vremena da pritisne taster tastature.
- Button press - koliko korisniku treba vremena da pritisne taster misa.
- Pointing - koliko korisniku treba da nesto ukaze, prevuce.
- Homing - koliko korisniku treba da prebaci ruku sa misa na tastaturu i obrnuto.
- Mental preparation - koliko korisniku treba da nesto uoci, vidi, priseti itd.
- Respoding - vreme cekanja korisnika na reakciju sistema - nema globalno definisano vreme

Treba istaci nekoliko stvari koje su vazne za date KLM operatore, ali i za KLM uopste. Pre svega, do razdvajanja keying i button press operatora, koji su manje vise isti, doslo je zbog drag-and-drop metode. Takodje, treba se istaci i to da se homing moze u potpunosti izbaciti (cemu se i tezi) ako se, recimo, kupe tastature koje u sebi imaju ugradjen mis ili se na tastaturi naprave neke precice. Precice se mogu definisati putem F2-F12 tastera na tastaturi, gde se opet iznad njih, na plastici, moze zapisati znacenje. U KLM model je uvedeno i nesto sto se zove starosni multiplekseri gde se vreme dodatno povecava u zavisnosti od starosnog doba korisnika. Ono sto je vazno istaci jeste da KLM ne podrzava greske, pretpostavlja se da korisnik ne gresi.

KLM, kao jedna od prediktivnih teorija, se u velikoj meri zasniva na predvidjanju kretanja korisnika kroz interfejs. Ako bi neki veci projekat u potpunosti bazirali na KLM-u najverovatnije bi se izgubili zbog veoma velikog broja mogucih scenarija. Zbog toga KLM u pomoc poziva GOMS, kao jos jednu od prediktivnih teorija. Treba istaci da je GOMS, osim sto je prediktivna, ujedno i eksplinatorna teorija. Eksplinatorna u smislu da objasnjava sta treba da se desi da bi se stiglo do nekog cilja, odnosno njegovih potciljeva (Goals), gde do svakog od tih potciljeva stizemo primenom nekih operatora/komandi (Operators), pri cemu te komande (operatori) mogu biti grupisane u metode (Methods). Koja metoda ce se izvrsiti zavisi od Selection rules. Princip formiranja GOMS modela jeste da se prvo definise neki krajnji cilj. Nakon sto se to odradi, pristupa se definisanju potciljeva koji vode ka ostvarivanju krajnjeg cilja. Kada se i to uradi, za svaki potcilj se definisu razlicite metode sa svojim skupom operatora koje mogu odvesti ka ostvarenju datog potcilja, pri cemu metode mogu biti birane na osnovu razlicitih uslova (Selection

rules). GOMS u stvari omogucava struktuiranje aktivnosti prilikom rada sa nekim softverom. Kao i kada je u pitanju KLM, tako i GOMS ima mnoge pretpostavke. Najveca pretpostavka (mana) mu je da covek ne gresi. Operacije u okviru GOMS-a mogu biti predstavljena na razlicitim nivoima. Ti nivou su sledeci:

- konceptualni nivo - najvisi nivo, sta se coveku desava u glavi
  - ako se operatori opisuju na konceptualnom nivou, onda je GOMS prava eksplinatorna teorija
- semanticki nivo
  - nivo komandi
- sintaksni nivou
  - nivo strukture komande
- leksickom nivou
  - nivo samog uredjaja i korisnika
  - treba postaviti prst, kliknuti, razmisliti itd.
  - svaki operator na leksickom nivou ima svog parnjaka u KLM-GOMS

Informaciona petlja predstavlja jedan krug kojim informacije putuje od korisnika ka masini. Uspesna informaciona petlja izmedju operatora i masine jeste da masina mora dati informacije o svom stanju u formi takvoj da operator preko svojih receptora (oci, usi, osecaj) primi informacije o stanju tog sistema. Ako primi informaciju o stanju sistema koje nije odgovarajuce, onda treba da pokrene svoje efektore, odnosno da izda naredbu koja ce popraviti dato stanje. Poenta jeste da dodje do spajanje efektora i receptora, a to je feedback. Kada efektori zadaju komandu, receptori odmah, istog trenutka, trebaju da dobiju feedback - povratnu informaciju da je sistem komandu primio.

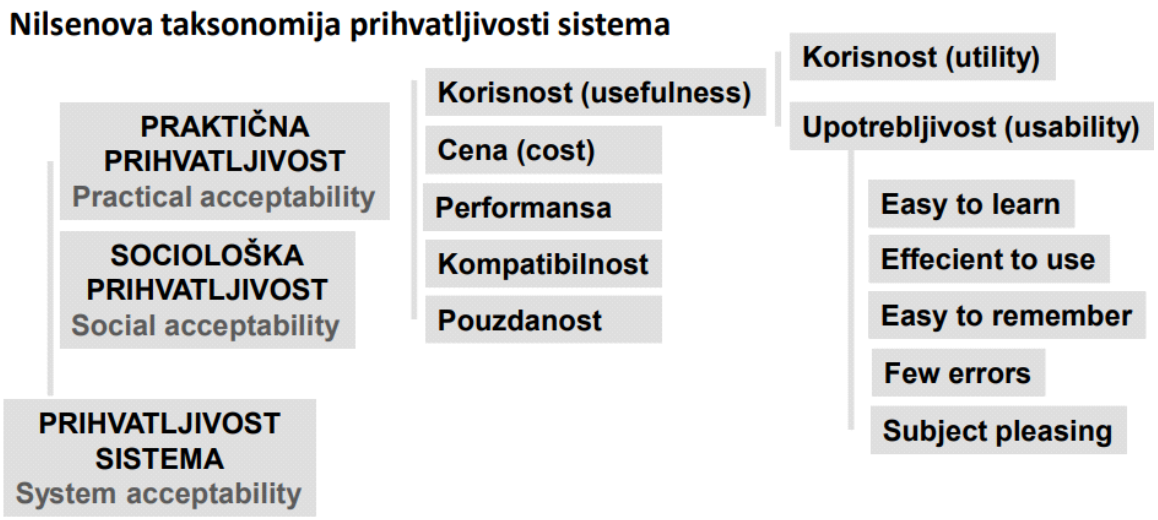
Informaciona petlja, takva kakva jeste, je izuzetno gruba i cesto se to primitivno shvatanje informacione petlje jos naziva i indrustrijska petlja. Don Norman je rasclanio informacionu petlju i time je priblizio programerima kako bi oni dalje mogli praviti softver koji je jos blizi korisniku. Prilikom njegovog definisanja informacione petlje, on nije koristio pojam efektora i receptora, kao njegove kolege sredinom 20 veka, vec pojmove kao sto su struja izvrsenja i struja evaluacije. Struja evaluacije predstavlja kretanje informacije od masine ka coveku, dok je struja izvrsenja u obrnutom smeru. Sada se postavlja pitanje: Kada je interfejs dobar? Kada nam struja evaluacije prikaze sve sto je potrebno, pritom i korisniku lako razumljivo, za potpuno shvatanje stanja sistema, onda je interfejs dobar. Ako se putem struje izvrsenja (egzekucije) definisu sve komande koje korisnik u datom trenutku zeli da zada masini, onda je interfejs dobar. Ukoliko dodje do nekog nepoklapanja, interfejs nije dobar. Da bi te struje bile dobre, samim tim i interakcija i interfejs, on je njih dodatno rasclanio. Da bi struja evaluacije bila uspesna, da bi korisnik jasno evaluirao u kom se stanju nalazi dati sistem, prvo sto trebamo da mu obezbedimo mi, kao tvorci interfejsa, jeste da bude u stanju da osmotri stanje sistema (skup parametara koji opisuju sistem u jednom trenutku). Kako se to obezbedjuje? Obezbedjuje se putem widgeta, pomocu kojih on moze da vidi vrednost jednog parametra, drugog parametra itd. Sledece sto trebamo da mu obezbedino jeste da moze izvrsiti interpretaciju stanja sistema. Ovo se odnosi na to da, recimo, ako se putem nekog widgeta definise temperatura, onda se treba, u zavisnosti od toga da li se sistem koristi u Americi ili Evropi, primenjivati F ili C kao merna jedinica. Ako je interfejs napravljen tako da se moze videti svako stanje sistema, pritom je obezbedjeno da korisnik na pravi nacin vrsi interpretaciju stanja, onda u evaluaciji ishoda korisnik samo sebi postavlja pitanje, moze se reci da predvidja, sta ce se desiti ako ostanu dati parametri u sistemu. Cesto se desava da i sam interfejs pomaze korisniku prilikom evaluacije ishoda. Ako se evaluacijom ishoda utvrdi da nesto nije u redu, onda korisnik prelazi u fazu formiranja cilja. Nakon sto se formira cilj krece struja izvrsenja. Struja egzekcuje pocinje sa formiranjem intencije. Formiranje intencije se bazira na odabiru namere, nacina izvrsenja prethodno formiranog cilja. Kada izabere nacin kojim ce izvrsiti dati cilj, korisnik vrsi specifikaciju akcije ili akcija kojima ce se izvrsiti izabrana namera. Poslednja vazna karika u datoj struji jeste izvrsenje datih akcija ili akcije u smislu da damo naredbe

racunaru.

Na razlicite nacine se moze utvrditi da ce interfejs prouzrokovati probleme. Pre svega, to se moze utvrditi prema zdravom razumu. U sustini, najveci procenat pronadjenih gresaka na interfejsu upravo i jesu od strane autora ili njegovih najblizih saradnika. Sledeca metoda jeste razvojem modela "human cognitive processing" putem kojeg se pokusavaju predvideti problemi koje ce korisnik imati prilikom koriscenja interfejsa. Jos jedan od nacina jeste putem razvoja teorije neuronske korelacije HCI koncepata. Razvoj se bazira na tome da se pred korisnika ili odabranu grupu korisnika (testera) postavi dati softver, zada im se koje aktivnosti trebaju da rade sa softverom, a pritom im se na gluvu stavi kapa putem koje se prati rad njihovog mozga. Prethodne dve metode su preuzete iz medicine i nisu bas mnogo korisni za nas jer nam pruzaju prilicno grube rezultate. Jedan od problema moze predstavljati i to sto je to ipak covek, pa prilikom rada sa softverom moze se desiti da mu zalutaju misli i seti se necega sto uopste nema veze sa nasim softverom. Poslednja metoda jeste putem testiranja interfejsa sa korisnicima. Data metoda je ujedno i najbolja, medjutim, korisnici moraju biti motivisani da bi vam stvarno ukazali na probleme sa kojim se softver potencijalno moze suociti.

U oblasti koja se bavi izucavanjem interakcije izmedju coveka i racunara postoji nekoliko bitnih ljudi. Medju njima je svakako i **Jakob Nilsen**. Prema Jakobu Nilsenu se prihvatljivost sistema bazira na prakticnoj i socioloskoj prihvatljivosti. Socioloska prihvatljivost je izuzetno bitna, sto se dosta puta dokazalo u praksi, i ona se bazira na tome koliko ce samo drustvo prihvatiti neki, odredjeni sistem. Primer se moze uzeti iz auto industrije. U pitanju je automobil Chevrolet Shevy Nova, koji je bio izuzetno popularan sirom sveta, osim u Spaniji, jer na spanskog jeziku "no va" znaci "ne ide". Prakticna prihvatljivost se stastoji iz nekoliko stvari. Prva je korisnost (usefulness). Korisnost u smislu lepeze funkcija koje dati sistem pruza, ali to nije dovoljno, jer te funkcije moraju biti ponudjene na taj nacin koji je korisniku blizak. Dalje, na prakticnu prihvatljivost utice i cena. Ako je nesto previse jeftino, postavlja se pitanje da li je to onda uopste dobro, a ako je preskupo, da li to stvarno vredi te pare. Na prakticnu prihvatljivost uticu i performanse sistema, u smislu koliko je vremena potrebno da se izvrsi neka funkcija. Kompatibilnost se odnosi na to da bi jako dobro bilo da dati sistem bude kompatibilim sa sistemima u okruzenju ili sistemom koji je vladao pre njega. Kompatibilnost je vazna sa stanovista toga da korisnik moze da izvrsi transfer znanja sa starije verzije na novu, odnosno ne mora u potpunosti, ispocetka, da uci o novom sistemu. Kao poslednje bitno za prakticnu prihvatljivost uzima se pouzdanost. Sto se tice prvog elementa koji cini prakticnu prihvatljivost, a to je korisnost (usefulness), ona se dalje sastoji od, opet,

korisnosti (utility) i upotrebljivosti (usability). Utility u smislu da postoje sve funkcije, da rade bez greska itd. Usability se odnosi na upotrebljivost datih utility. Upotrebljivost se opet sastoji iz nekoliko stvari. Pre svega da je "easy to learn", da se lako moze nauciti. Da je "effective to use", da nije zamarajuci prilikom koriscenja. Takodje, "easy to remeber", da ne mora da se pamti nista prilikom koriscenja sistema, tacnije da sistem pruza nagovestaje sta dalje treba odraditi. Ako vec ima gresaka, treba obezbediti da ih ima malo. To je sledeca stvar u okviru upotrebljivosti. Poslednje jeste "subject pleasing", u smislu da je korisnik zadovoljan nakon koriscenja datog sistema. **Uvek je bolje da 30% korisnika bude zadovoljno 100%, nego 100% korisnika da bude 30%!**



Postoje razlicite metode putem kojih eksperti mogu da testiraju upotrebljivost (usability) nekon proizvoda. Prva jeste evaluacija po heuristikama. Ako se za heuristike kazu da su saveti koji dolaze od ljudi kojima se treba verovati, onda se za prvu metodu moze reci da predstavlja ispitivanje da li proizvodi zadovoljavaju te savete. U praksi se pokazalo da ako se ne drzimo tih saveta sigurno necemo uspeti, ali to ne znaci da cemo sigurno uspeti ako ih se pridrzavamo. Najbolje je, kada se koristi dati metod, angazovati tim od 3 do 5 ekperata koji ce ispitivati proizvod po izabranih heuristikama. Smatra se da 5 angazovanih eksperata mogu otkriti cak do 75% nepoklapanja. Treba istaci i to da angazovanje veceg broja eksperata znaci i veci broj misljenje, gde se postavlja pitanje koji je od eksperata onda u pravu. Sledeca metoda je revizija po smernicama. Razlika izmedju evaluacije po heuristikama i revizije po smernicama se ogleda u tome sto smernice uvek dolaze od strane industrije. Industrija daje smernice kako nesto treba napraviti. Smernice su vrlo konkretne, za razliku od heuristika koje su vise filozofsko, apstraktno misljenje o izgledu proizvoda. Smernice je lako proveriti, ali ih je brojcano jako puno. Inspekcija konzistentnosti je sledeca metoda. Data metoda se bazira na tome da ekspert uzima nas proizvod na razmatranje i gleda gde se pojavila nekonzistentnost. Nekonzistentnost se odnosi na neko neslaganje. Neslaganje po bilo cemu. To moze biti boja, terminologija, format teksta itd. Ispitivanje nekonzistentnosti potpada, manje-vise, pod svaku heuristika, ali je izdvojena i kao posebna tehnika ispitivanja usability (upotrebljivosti) proizvoda kada nemamo para da platimo eksperta za isptivanje po heuristikama. Naredna metoda je "cognitive walkthought". Eksperti gurnu svoje znanje sa strane i ponasaju se kao neiskusni korisnici. Formalna inspekcija utilitarnosti predstavlja ispitivanje upotrebljivosti nekog proizovda unutar kuce koja razvija taj proizvod. Ekspert predstavlja lice iz kuce proizvodioca softvera koji se ponasa kao sudija, dok je razvojni tim u ulozi odbrane. Ulogu tuzioca ima drugi razvojni tim iz iste kuce, koji nije radio na datom softveru. Ekspert nije taj koji odlucuje ko je u pravu, vec sluzi samo da pomogne da se ode, konvergira ka nekom resenju. Minus metode jeste to sto cela kuca staje sa radom, dok je velika prednost to sto pocetnici, koji

su obavezno ukljuceni u rasprave, kroz njih brzo uce. Poslednja i najskuplja tehnika jeste usability labs. Bazira se na tome da se na nekom mestu postavi puno soba, gde u svakoj postoje racunari i pritom se svaka soba moze kontrolisati. Pod kontrolisati misli se na to da se moze definisati temperatura u sobi, vlaga, zvuk itd. Sve se snima i u centru laboratorije (znaci izmedju svih tih soba) postoji jedan centralni zid, od stakla, koji ima pogled ka svim tim sobama. U tom centru se nalaze razliciti psiholozi, dizajneri, lekari itd. koji prate situaciju i ponasnje korisnika. Prethodno iznesena metoda je najbolja, ali izuzetno skupa.

Shneiderman-ovih osam zlatnih pravila:

1. teziti konzistentnosti
2. omoguciti frekventnijim korisnicima upotrebu precica
3. davati informativni feedback
4. projektovati dijaloge naglasene zatvorenosti
   - grupisanje akcija da imaju jasan pocetak i kraj
5. ponuditi prevenciju i rukovanje greskom
   - obezbediti da korisnik ne napravi gresku, a i ako se desi, da ga izvucemo sto pre i sto bolje
6. dozvoliti ponistenje efekata akcije (undo)
   - veliko je olaksenje kod pocetnika
   - omogucava korisnicima da istrazuju softver
   - cak i ako se zaluta, moze se vratiti na poslednje mesto koje je bilo poznato
7. interno podrzavati kontrolu
   - trebamo obezbediti osecaj kod korisnika da ima kontrolu
   - trebamo mu pomoci u mnogome, ali ne toliko da softver sve radi umesto njega
8. redukovati opterecenja radne memorije
   - 7 +- 2 ranije, danas cak 11
   - u svakom trenutku maksimalno 11 informacija
   - treba se napraviti softver takav da on navodi coveka ka necemo, a ne da covek to vadi iz malog mozga

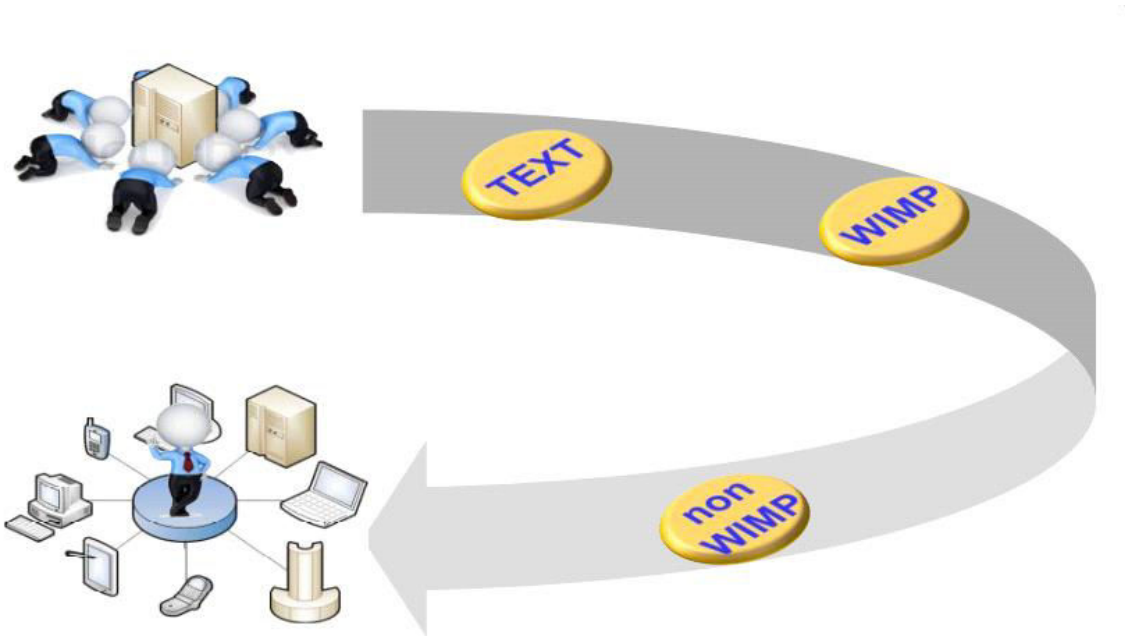
Nilsenovi principi (vise orijentisano ka internet sajtovima):

1. nalikovati stvarnosti
   - kako vizuelno tako i terminologijom
2. konzistentnsot i standardi
3. help i dokumentacija
   - online help (poentiranjem, stavljanjem misa, na neki deo interfejsa se dobija kratka informacija o tome delu
   - offline help (klikom na odredjeni deo interfejsa, a zatim i f1, gde se otvara novi prozor)
4. korisnikova kontrola i sloboda
   - undo
   - cancel
5. vidljiv status sistema
   - korisnik uvek moze da vidi sta se desava na sajtu
   - svestan stanja
   - promeni se izgled kursosa prilikom cekanja, postavi se neki "progress bar" itd.
6. fleksibilnost i efikasnost
   - korisnik moze da prilagodjava okruzenje svoj potrebama
   - softver je brz
7. prevencija gresaka
8. prepoznaj, ne da se pamti

- redukcija radne memorije korisnika

9. prijava greske, dijagnostika i oporavak
10. estetican i minimalisticki dizajn

Na pocetku razvoja racunara, ali i racunarstva kao naucne oblasti, ljudi su bili ti koji su vise okrenuti ka racunarima i oni su ti koji su se divili kako jedna, tada izuzetno velika masina, moze umesto njih da odradjuje neki posao. Ti poslovi, za danasnje standardne, nisu nista veliko. To su najcesce bili neki matematicki proracuni. Medjutim, cak i ti matematicki proracuni su kod svakog coveka budili divljenje, jer masina kroz samo delimicnu covekovu kontolu, moze da odradi neki posao. Primer takvog racunara jeste Eniac, koji je razvijan sredinom 40-tih godina. Prvim racunarima su se cak komande zadavale fizickim putem, tako sto se vrsi prebacivanje metalnih plocicica sa jednog na drugo mesto ili pak prevezivanjem razlicih kablova. Medjutim, to je ubrzo prevazidjeno i preslo se na zadavanje komandi u tekstualnom obliku. Takodje, ni kod njih nije postojao GUI (graficki korisnicki interfejs). Kako su se racunari razvijali, tako su i ljudi sve manje poceli da primecuju prisutnost racunara iako je njihov znacaj bio sve veci. Racunari su poceli da se shvataju kao nesto sto je tu zbog njih, da njima olaksa zivot i prestalo je ono prvobitno divljenje. Njihova prisutnost se pocela primecivati tek kada bi prestali raditi. Promenio se i unos komandi. Unos komandi se poceo bazirati na otvaranju razlicitih prozora, kliktanju ikonica, poentiraju itd (WIMP). Koriscenje se svelo na korisnikovo kretanje kroz virtuelni svet datog racunara. Na kraju se doslo do toga da covek pocinje upravljati racunarom putem pokreta ruku, nogu, uopste gestikalacijom, ali i glasom (NON-WIMP). Mnogi danasnji racunari su cak tako isprogramirani da im ljudski faktor nije ni potreban, vec na osnovu razlicitih senzora mogu sami pokretati odgovarajuce procedure. Programiranje takvih racunara je jedan izuzetno tezak postupak, ali itekako isplativ.



Programiranje text interfejsa se radi kada je imperativ, fokus na naredbama koje sistem pruza, a ne na lepom i funkcionalnom korisnickom interfejsu sa stanovista HCI. Prednost text interfejsa jeste ta sto se komande lako citaju i razumeju, dok je veliki nedostatak ceste greske koje mogu nastati prilikom njihovog izdavanja (pisanja).

Postoji nekoliko vaznih osoba kada je HCI, odnosno interakcija izmedju coveka i racunara, u pitanju. Jedna od vazniji osoba je svakako Ajvan Saderlend. Treba odmah istaci da je to osoba koja se

smatra ocem modernog HCI-a. Osmislio je koncepte danasnjeg modernog interfejsa. Za SketchPad, koji je njegov izum iz 1962. godine, se smatra da predstavlja prethodnika danasnjih grafickih korisnickih interfejsa. Uveo je ikonice i transformisao tekstualno orijentisane interfejse u objektne. Vazan je i sa stanovista hardvera. Prvi je poceo da razvija "low-cost" graficke terminale. Sledeca vazna osoba je Daglas Engelbart. Neki od vaznijih njegovih izuma su mis i stenografska tastatura. Stenografska tastatura je uredjaj koji omogucava da se pomocu jednog karaktera / tastera zapise cela rec ili cak recenica. Izuzetno koristan uredjaj kada je zapis govora u pitanju (sudnica). Takodje je uveo visestruke prozore i prikaze visoke rezolucije. Bio je jedan od clanova tima koji su razvijali ARPANET i sistem elektronske poste. Jos jedna bitna osoba je Alan Kej koji je definisao ideju za lap-top i tablet racunare. Na kraju ne treba zaboraviti i Scheidermana koji je definisao "direktnu manipulaciju". Objasnio je osnovne ideje korisnikove kontrole, a to su: vidljivost objekata i akcija, potom brze, reverzibilne i inkrementalne akcije, kao i zamena kompleksne command-line sintakse direktnom, vizuelnom manipulacijom objekata od interasa. Sve prethodno napisano je usko vezano za graficke korisnicke interfejse koje se bazira na WIMP programiranju.

Kao sto je ranije receno, poslednji oblik zadavanja komandi racunaru, koje je danas sve prisutnije, jeste putem gestikulacije i govora. Takvo programiranje je jos poznato i kao nonWIMP. Za razliku od GUI-a koji pruza serijsku i diskretnu interakciju, nonWIMP obezbedjuje paralelnu i kontinualnu interakciju.

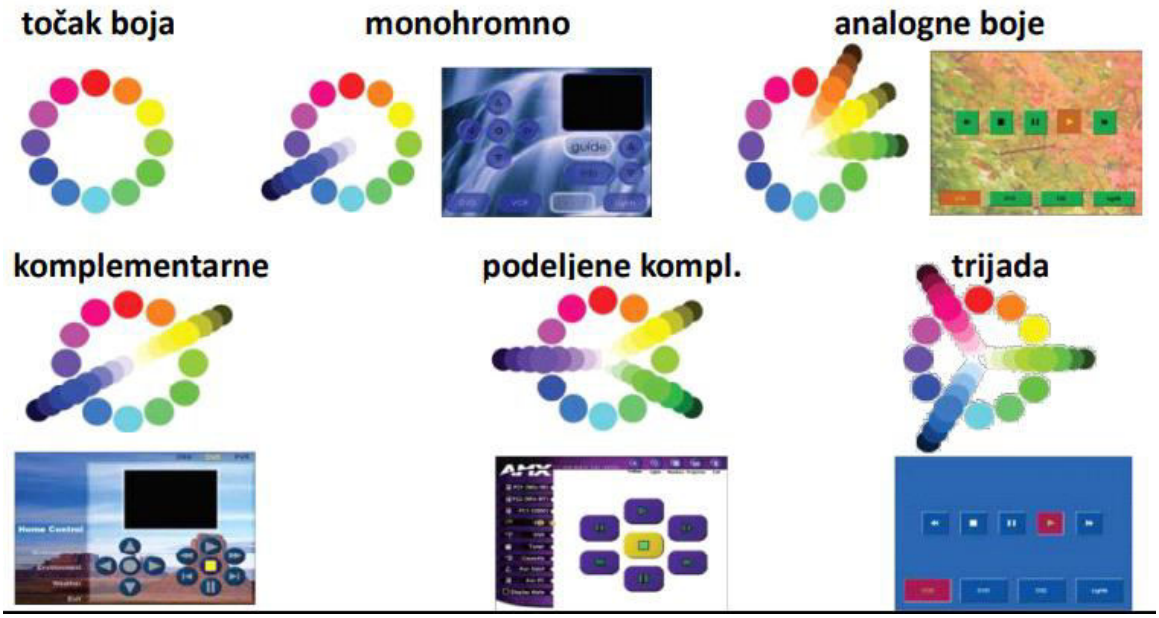
Iako je receno da je poslednji oblik zadavanja komandi, a samim tim i interakcije sa racunarom, preko pokreta i glasa, treba istaci i jedan koji je trenutno u razvoju. Rec je o interakciji baziranoj na senzorima, gde korisnik manje-vise nije ni svestan postojanja interakcije. Ta interakcija je implicitna, jer sistem bukvalno predvidja sta je korisniku potrebno. Treba istaci i to da implicitna interakcija ne cini HCI besmislenim jer i tada treba racunare osposobiti da "osete i razumeju" sta korisnik zahteva na osnovu svojih aktivnosti.

Osnovni nosioci informacija na korisnickom interfejsu, odnosno osnovni mediji za prenos sadrzaja ka korisniku, mogu se podeliti u dve velike grupe. Prva je diskretna, druga je kontialna. Diskretne tehnike, mediji, nosioci informacija su oni koji ponude informacije i korisnik ih moze tumaciti danas, sutra , prekosutra, bilo kada. U ovu grupu spadaju tekst i slika. Za razliku od diskretnih, kontinualni mediji u velikoj meri zavise od vremena. To znaci da kada nam se informacije prenose preko ovakvih medija, sam prenos pracenja toka infromacija dovodi do nerazumevanja kompletne poruke. U date medije spadaju video, zvuk i animacije. Vrlo je cesta kombinacija kontinualnih i diskretnih medija. Kada dodje do takve kombinacije, novo-nastali medij se zove multimedija.

Pomocu teksta, kao diskretnog medija, se najlakse moze prezentovati neka informacija. Medjutim, ta informacija, da bi mogla biti prezentovana pomocu teksta, po prirodi mora biti nezavisna od vremena. Pritom, treba se odmah istaci da se tekst koristi za opis detalja, za detaljisanje, za razliku od slike koja pruza coveku samo uvid u sadrzaj. Najbitnije za tekst koji se stavlja u interfejs jeste da bude citljiv. Cime postici citljivost teksta? Dva atributa fonta najvise uticu na to. Prvi je velicina fonta. Sto je korisnik stariji, to mu treba veci font. Drugi se odnosi na boju i kontrast u odnosu na pozadinu. Cesto se desava da je velicina teksta korektna, ali je kontrast izmedju boje slova i boje povrsine izuzetno mala, pa je samim tim necitljivost velika. Opet treba istaci da sto je osoba starija, pored veceg fonta, zahteva i veci kontrast. Pored ovih, glavnih atributa, postoje dodatni koji takodje uticu na citljivost. Jedan od dodatnih atributa jeste poravnanje. Tekst uvek treba biti poravnat uz stranu od koje zapocinjemo citanje (ili obostrano). U nasem slucaju to je uz levu ivicu. Centralno poravnanje iskljucivo koristiti za naslove. Sledeci atribut jeste sirina linije. Ona treba biti taman toliko da se covek ne umara. Prored izmedju teksta treba napraviti tako da se slova iz susednih redova medjusobno ne dodiruju. Pritom, treba paziti da

prored ne bude previse velik, pa da korisnik ne shvati odvojene redove kao razlicite pasuse. Citljivosti doprinusi u velikoj meri i uvlacenje pasusa. Na kraju, treba ukratko reci nesto i o naglasavanju teksta. Postoje dve velike grupe fontova, to su serif i sans serif. U serif spadaju oni fontovi cija slova imaju kvacice, i takav font se koristi kada zeli nesto da se naglasi, da upadne u oci. Sans serif je vise namenjen za ostatak teksta, jer je relaksirajuci. Takodje, Serif se iskljucivo koristi kada je veca rezolucija ekrana, jer pri manjoj rezoluciji kvacice mogu da uticu negativno na citljivost. Kod stampaca se moze koristiti i jedno i drugo. Na koji nacin covek cita? Svaka rec se sastoji od ljuske ("bamp"). Ljuska jeste linija oko reci. Covek kada cita neku rec koju zna, onda je jednostavno procita odjednom, prepozna njenu ljusku. Pritom ni ne gleda slova unutra, zbog cega se cesto desava da ljudi ni ne prepoznaju slovne greske. Kada naidje na rec koju ne zna, onda vrsi citanje slovo po slovo. Ako opet naidje na tu rec, opet ce ici slovo po slovo i tako nekoliko puta. Na kraju ce se to pretvoriti u jednostavno prepoznavanje njenog "bump"-a. Citanje velikih slova je zamorno jer je "bump" takvih reci uvek isti (pravougaonik). Isticanje reci je bolje zato raditi preko promene fonta. Recimo koristiti Serif fontove. Podvlacenje se izbegava pri isticanju, narocito u Internet tehnologijama, zbog moguceg mesanja sa linkovima. Za naglasavanje se moze koristiti i bold, jer vrsi dobro povecanje kontrasta.

Boje nisu medij pomocu kojeg se vrsi prenos neke informacije, barem ne direktno, ali su izuzetno vazne kada je interfejs u pitanju. Pomocu njih se vrsi naglasavanje odredjenih elemenata interfejsa, ali i skretanje paznje. Treba biti vrlo oprezan kada je koriscenje boja u pitanju i ne treba ceo interfejs zasnivati na njima, jer se uvek treba u vidu imati da postoje ljudi koji ne mogu da razlikuju boje (razliciti oblici daltonizma). Takodje, kada se interfejs pravi za odredjenu struku gde su boje izuzetno vazne, onda se takvi ljudi stvaljaju sa strane. Kao sto je receno, boje su izuzetno dobre da privuku paznju, ali vremenom pocinju da smetaju. Preporuka jeste da u softveru koristi od 4 do 7 boja. Vrlo cesto se koriste i za grupisanje razlicitih elemenata interfejsa. Postoji nekoliko tehnika za koriscenje boja. Te tehnike su prikazane na slici ispod. Boje, pre svega, mozemo koristiti monohromno. To znaci da koristi jedna boja i razlicite njene nijanse u odnosu na kolicinu svetlosti. Korisnicima ce to biti lepo na oko, ali ce kreatorima interfejsa biti jako tesko da iskazu razlicita stanja widgeta na interfejsu. Ako boje koristimo analogno, to znaci da koristimo vise boja u jednom delu spektra boja. Komplementarno koriscenje boja se bazira na tome da imamo dve suprotne boje. Lako je za prikazati razlicita stanja interfejsa. Medjutim, mana jeste sto tada interfejs vise nije toliko smirujuci. Ostale tehnike jesu podeljena komplementarnost i trijada.



Sliku koristimo da iskazemo nesto, kada se kaze nesto misli se na neku informaciju, koja se sastoji iz grupe podinformacija koje su vrlo usko povezane i koje samo zajedno daju kompletnu predstavu necega. Slika u stvari sluzi da korisniku da uvid u neke slozene informacije. Iako je ranije receno da se tekst koristi za detaljisanje, postoje neke situacije kada bi slika mnogo lepse objasnila odredjenu situaciju od teksta. Pored toga koristi se da da neki pregled situacije, odnosno "pre-view". Problem sa slikom jeste taj sto treba znati putem slike predstaviti tu neku stvar. Kada se to zna, onda ona predstavlja jedno izuzetno dobro i efikasno resenje. Jos jedan problem jeste taj sto svako sliku vidi drugacije, samim tim je i tumaci na drugi nacin. Glavne elemente svi uglavnom vide isto. Problem su oni sporedni. Cak ni ista osoba ne vidi sporedne detalje slike pri svakom pogledu isto. Kao i kada je u pitanju tekst, slika se koristi kada informacije nisu zavisne od vremena.

Razlika izmedju videa i animacije, kao kontinualnih medija, se ogleda u tome sto je video uvek uzorak stvarnosti. Animacija moze biti predstava realnost, ali se cesce koristi za predstavljanje neke fantazije, apstrakcije. Na primer, letece automobile, dinosauruse itd. Animacija se moze uvek koristiti, samo je pitanje da li se moze na pravi nacin izvrsiti animiranje zeljenog. Prednost animacije jeste i ta sto moze da iskljuci neke stvari koje nam nisu od interesa (apstrahcija), a u videu bi se eventualno pojavili. Najcesca primena videa jeste kada je potrebno nauciti coveka da ovlada nekom vestinom, a da je pritom to tesko odraditi putem teksta ili slike. Naravno i animacije se mogu koristiti za to, ali je pomocu videa to jednostavnije realizovati.

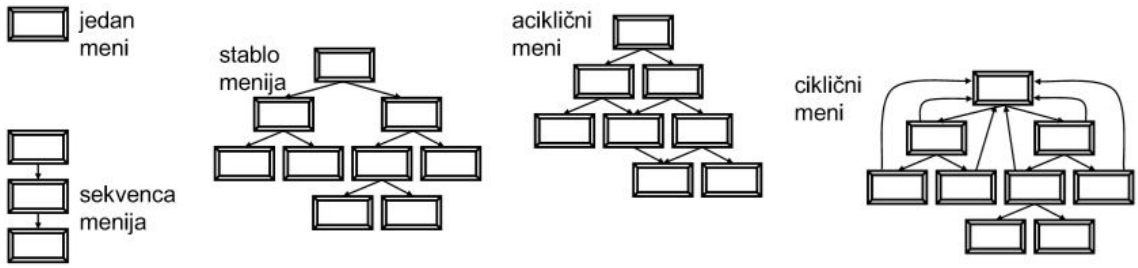
Dok je vid dominantno culo za obradu informacija, culo sluha je dominantno sto se tice zastite. Lepezu informacija koju nam daju oci dopunjuju ostala cula, pa tako i culo sluha. Pri koriscenju zvuka treba biti skrt. Razlog lezi u tome sto ako budemo koristili zvuk kao feedback za razlicite stvari, korisnik ce se navici ili jos gore, isljucice ga. Najcesce se koristi za upozorenje. Za to se koriste kratki, periodni i veoma iritirajuci zvukovi. Medjutim, ne mora mu to biti jedina namena. Moze biti i neki mio, kratak zvuk da nas obavesti da je stigla poruka ili na primer signal da je neka dugacka radnja zavrsena. Treca uloga zvuka jeste, kada je ujedno i duzi, dopuna onoga sto korisnik konzumira drugim culima. Primer bi bio web prezentacija muzeja gde se gledaju slike iz srednje veka, a onda uz to ide i srednjovekovna muzika. Kaze se da je takav zvuk ambijentalan.

Stil interakcije se odnosi na nacin komunikacije korisnika sa interfejsom. Koji od stilova ce korisnik koristiti zavisi od zadatka koji treba da se resi, ali i tehnicke podrske (uredjaji koje koristimo pri interakciji). Prilikom rada softvera, koriste se razlicite kombinacije datih stilova. Nacin na koji uredjaj koristimo prilikom te interakcije zove se interakciona tehnika. Neki uredjaji imaju razlicite tehnike, odnosno koriste se na razlicite nacine, prema tome imaju vise interakcionih tehnika putem kojih moze da se realizuje razliciti stilovi. Takodje, neki stil interakcije moze biti obezbedjen na razlicite nacine, putem razlicitih interakcionih tenika jednog ili vise uredjaja. Interakcioni stilovi su sledeci: komandna linija, meni, forme, direktna manipulacija, i antropomorfni.

Komanda linija obezbedjuje najstariji nacin interakcije. Izuzetno je tezak za koriscenje, u smislu da je zamoran, jer korisnik mora dobro poznavati sintaksu i semantiku. Potencijalno cak mora i pamtiti veliku kolicinu naredbi koju sistem poseduje. Komanda linija se ne mora uvek odnositi na ukucanu komandu, vec moze i na izgovorenu. Sa strane kreatora interfejsa, ona takodje nije bas jednostavna za implemntaciju zbog silnih sintaksnih, semantickih i leksickih provera unosa. S njom mozemo maltene sve obezbediti. Komanda linija je najbolja kada je potrebna tacnost. Ima "homing" problem, jer nam je cesto ruka na pointerskom uredjaju pa je potrebno prebacivanje na tastaturu.

Meni se koristi radi obezbedjivanja izbora. Obezbedjuje da se iz skupa stavki izabere ona koja u

tom trenutku najvise odgovara. Da bi korisnik mogao da izabere pravu stavku, one moraju biti jasno deskriptovane, objasnjene sta znaci, ali i da se po imenu razlikuju od drugih stavnik, da se korisnik ne dovede u poziciji premisljanja. U HCI-u se pod pojmom meni podrazumeva bilo koji vid izbora. Losa stvar kod menija se ogleda u tome sto je tezak za implementaciju u smislu dizajna. Sta raditi kada ima veliki broj stavki, na koji nacin to organizovati? To se moze odraditi dekomponovanjem, tacnije grupisanjem stavki u odredjene potskupove, gde se onda izmedju njih umece neka linija ili jednostavno razgranavanje. Takodje, postavlja se i pitanje na koji nacin redosledno ih organizovati. To se moze ostvariti na razlicite nacine (alfabetski, prema frekvenciji upotrebe itd.). Kada je u pitanju implementica tehnickog dela, tu je veoma jednostavan. Postoje razlicite organizacije datog stila interakcije. Ti stilovi su predstavljeni slikom ispod.



Meni se mogu napraviti na razlicite nacine. Jednostruki ili binarni meni jeste najjednostavnija realizacija. Bazira se na izboru izmedju dve moguce stavke. Sledeci nacin jeste n-arni meni. Postoji dva tipa n-arnih menija, disjunktivni i konjuktivni. Disjunktivni se bazira na tome da se iz skupa stavki izabere samo jedna, dok su konjuktivni bazirani na izboru veceg broja. Za realizaciju disjunktivnih meni-a se moze koristiti radio button ili drop lista. Raddio button je jednostavniji za koriscenje, ali drop lista zauzima manje prostora. Takodje, radio button daje odmah taj neki pre-view. Smatra se da do 8-10 stavki treba koristiti radio button, za sve preko toga preci na drop listu. Realizacija konjuktivnih je uglavnom bazirana na check boxovima. Pull-down meni se bazira na tome da imamo jedan osnovni skup stavki predstavljen preko trake, a onda klikom ili pomeranjem misa na datu stavku dolazi do otvaranja novog prozora ka dole ili izvrsenja funkcije. Pop-up meni jeste onaj koji se pojavljaju na klik. Do otvaranja najcesce dolazi u desnom matematickom kvadratnu u odnosu na izvrsen klik. Razlog lezi u tome sto je vecina ljudi desnoruka pa im je veoma lako onda da izvrse prevlacenje rukom u desno. Vrlo je lako doci do prve stavke tog menia koji iskoci. Ne moze se isto reci i za poslednju. Iz toga razloga su se poceli praviti iskacuci meni koji su kruznog oblika (pop-up pie), tako da je do svake stavke mnogo lakse doci. Problem lezi u tome sto smo tada ograniceni sa brojem stavki koje se mogu ponuditi. Sto je veci broj stavki u datoj realizaciji, to je korisniku teze da je klikne. Sledeci jeste skrolovani 2D. Izbori se sastoje iz dve komponente, to su ime i slika. Veoma je nepregledan i tezak, mucan za koriscenje, ali u odredjenim situacijama, na primer kada se gleda sadrzaj nekog direktorijuma, je neizbezan. Jos neki od menia jesu linkovi, alfaklizaci, ikonice itd. Kod ikonica izazov jeste to sto treba naci odgovarajucu sliku koja reprezentuje datu akciju, ali barem nema problema sa tekstom. Takodje, moze postojati i visestruki meni koji se onda stavlja u okviru jednog objekta i gde se putem prethodnih metoda se mogu realizovati razliciti izbori. Cesto se takva serija medjusobno zavisnih menija naziva i dijalog. Izazov lezi u tome kako na pravi nacin voditi korisnika kroz dati dijalog. Ako imamo stablo menija, treba se vrsiti takvo imenovanje da korisnik citanjem izbora dobije jednu jasnu recenicu. Pozadina menia treba biti netransparentna, a ako vec jeste onda da je do 50%. Nekada je vazna ta transparetnost, jer se u pozadini moze nesto desavati sto nam je vazno kada je izbor u meniju u pitanju. Na kraju, treba istaci da postoje razlicite metode ubrzanja kretanja kroz meni. Typeahead se bazira na tome da korisnik jednostavno unosi komandu koja reprezentuje kretanje kroz meni. Nedostatak jeste taj sto ne postoji echo. Nece komanda biti prikazana na ekranu, potpuno slepo kucanje. Indeksiranje stavki je sledeca metoda i ona se

bazira na tome da stavkama dodelimo neke numericke oznake, a potom unosenjem sekvence nekih brojeva izvrsimo odredjeni izbor. Poslednja tehnika jesu makroi. Oni se baziraju na tome da sam korisnik izvrsi neko povezivanje stavki i definisanje izbora.

Forme sluze za unosenje nekih atributa koji su vazni radi pravilnog izvrsenja neke akcije. Proistekle su iz realnog zivota i postojanja papiranih formulara za popunjavanje. Treba da ima svoj naslov koji ce u nekoliko reci definisati celu sustinu, svrhu date forme. Takodje, treba forma da da razumljive smernice kako da korisnik nesto popuni (staviti naziv pored mesta za unos). Pritom ta polja trebaju biti logicki grupisani. Teznja konzistentnosti, vizuelno privlacna, prevencija i rukovanje greskom, hint poruke o polju, signal da je moguc zavrsetak unosa, definisanje duzine polja su samo neke od stvari koji dobrinose dobroj formi.

Direktna manipulacija se zasniva na tome da korisnik vrsi direktnu manipulaciju prikazima, sadrzajem interfejsa. Direktna manipulacija je proistekla iz zelje da se korisnik "skine" sa kucanja po tastaturi, danas i po ekranu, nego da koristi mis ili neki drugi pointerski uredjaj. Jedan od oceva direktne manipulacije jeste Shnaiderman. Sto je meta bliza, to je potreba za njenu velicinu manja, dok sa povecanjem njene udaljenosti raste i potreba za velicinom. Najveca prednost direktne manipulacije se ogleda u tome sto omogucava direktnu predstavu zadataka, pa je prema tome korisniku izuzetno jednostavna, ali i zabavna za koriscenje. Omogucava mu transfer znanja sa pravih, fizickih uredjaja na softverske reprezentacije istih. Nedostaci jesu u tome sto su takvi softveri teski za programiranje a i nisu najbolje resenje kada imamo malo prostora za graficke prikaze. Za direktnu manipulaciju, kao stil interakcije, su vazne metafore, odnosno da se korisniku putem necega jednostavije definise nesto drugo. Metofe trebaju biti takve da coveku uvek bude jasno sta predstavlaju, sta simbolikuju. Najcesce se za tu primenu, a i uopste za direktnu manipuaciju, koriste ikonice, gde njihov izgled definise koju akciju mogu da izazovu. Direktno manipulacija se cesto nalazi i okviru nekog dijaloga (serija menija). Treba istaci da dijalog u stvari predstavlja seriju menija koji se kombinuju sa formama i direktnom manipulacijom. Sluze da korisnik u okviru jednog prozora razmeni puno informacija sa racunarom. Gestikulacija takodje predstavlja jedan od oblika direktne manipulacije, gde se sada manipulacija ne radi pomocu pointera vec putem ruku, prstiju. Problem sa gestikulacijom jeste taj sto nece covek svaki put na isti nacin uraditi neki pokret, tako da se masina treba napraviti da moze da prepozna i te razlicite oblike, verzije jednog pokreta.

Antropomorfni stil predstavlja onaj stil interakcije koji je slican interakciji, komunikaciji izmedju dva coveka. U stvari, antropomorfni stil predstavlja gestikulaciju kojoj je pridodat govor.

Dizajnerski prostor jeste prostor kroz koji se dizajner krece i obavlja funkciju dizajniranja interfejsa. Za tu namenu su mu potrebni widgeti-i. Widget-i su sve ono sto korisnik moze da vidi na ekranu. Dizajner moze koristiti vec postojece widgete ili sam praviti nove. Medjutim, prilikom dizajniranja odredjenog interfejsa nisu widget-i jedina stvar od kojih zavisi dizajn.

Dizajnerski prostori interakcije su prostori u kome se krece dizajner i dizajnira interfejs, gde on naravno zna za koju grupu korisnika vrsi dizajniranje, a nakon toga racunarci preuzimaju taj dizaj i implementiraju ga. Da bi dizajner napravio odgovarajuci dizajn, njemu dizajnerski prostor mora obezbediti widgete. Widgeti se najlakse mogu definisati kao vizuelne predstave. Kada ih ima ili kada moze napraviti nove, onda je u stanju da na pravi nacin prenese metaforu koriscenjem odgovarajuceg widgeta. Dizajner ce zavrsiti taj "dizajnerski posao" koji ce omoguciti vizelni prikaz razlicitih stvari, medjutim korisnicima treba omoguciti interakciju sa tim vizuelnim predstavama (widgetima). Za to sluze interakcioni uredjaji. Pomocu uredjaja korisnici pristupaju widgeti, a onda softver to treba da protumaci kao odredjenu naredbu. Kao sto je ranije receno, interakcione uredjaje je moguce koristiti na razlicite

nacine. Razlicitino koriscenje jednog interakcionog uredjaja naziva interakcione tehnike. To znaci da se putem razlicitih koriscenja nekog uredjaje moze pristupati nekom widgetu. Moze se reci da dizajnerski prostor interakcije cine widgeti, interakcioni uredjaji i interakcione tehnike. Imamo nekoliko dizajnerskih prostora interakcije. To su GUI, WEB stranice, mobilni uredjaji, ugradni sistemi i sveprisutno racunarstvo.

Graficki korisnicki interfejs, poznatiji jos i kao GUI, jeste dizajnerski prostor koji je usko vezan za odredjenu masinu i izuzetno zavisi od operativnog sistema. Moze se reci da je usko povezan ("dedicated") za odredjenu hardversko-softversku kombinaciju. Cak se cesto desava da je vezan samo za jednu verziju operativnog sistema. Dobro kod GUI-a jeste to sto su dizajnerima vrata otvorena da naprave dizajn kakav moze najbolje odgovarati korisnicima. To moze da se ostvari jer su dati widgeti usko povezani sa operativnim sistemom, odnosno on je taj koji direktno podrzava svako pojavljivanje, ponasanje i tipove interakcije sa datim widgetom. Tu je i najveca mana, a to je da su onda takva dizajnerska resenja usko vezana za datu platformu. Prenosivost je bukvalno nikakva. Da li je potrebno ulagati puno truda u nesto sto se moze iskoristiti samo u specificnim situacijama jeste najvece pitanje. Dobar GUI se bazira na sledecim stvarima. Simplicizam. Bolje izbaciti ono sto treba, nego ostaviti nesto sto nije potrebno! Sto je interfejs jednostavniji, to ce se naci veci krug ljudi koji ce se u njemu pronaci i za koje ce dato resenje biti perfektno. Sledece je kontrast. Kontrast po boji, oblicima, polozaju itd. Potom, bela margina. Omogucava separaciju. Korisniku je lako da razdvoji odredjene delove interfejsa putem beline. Mnogo je lepse nego neke linije. Takodje, tu su balans i poravnanje. Balans u smislu broja widgeta, boje, oblika itd. Postoji nekoliko nacina da se obezbedi simplicizam. Prvi jeste putem redukcije, odnosno izbacivanja. Pravilnost je jos jedan od nacina. Pravilnost doprinosi simplicizmu tako sto nece dovesti do smanjenja broja widgeta, ali pravilnom upotrebom istih (lep raspored, oblik, boja) moze coveku da stvori privid jednostavnosti. Na kraju, simplicizam se moze ostvariti i putem "double-duty". Ono je doslo iz Web-a i bazira se na tome da se vise stvari grupise u jedan widget. Na primer smestanje hint-a u okviru polja za tekstualni unos. Bertin je definisao 7 vizuelnih promenljivih koje su izuzetno znacajne sa stanovista GUI-a. Te promenljive su boja, oblik, orijentacija, pozicija, tekstutura, velicina i vrednost (u smislu osvetljenosti). Igranjem vizuelnim promenljivama moze se na mocan nacin uticati na korisnika da u dizajnu interfejsa veoma lako dodje do nekih zakljucaka, nekog shvatanja itd. Tu se sada dolazi do necega sto se zove selektivna i asocijativna percepcija. U selektivne promenljive spadaju sve osim oblika, dok u asocijativne spadaju sve osim oblika i vrednosti. Takodje, lepo bi bilo spomenuti i Gestalt teoriju. Ona se bazira na tome da covek veoma lako prepoznaje slicnost, blizinu, kontinualnost i zatvorenost. Da bi se obezbedilo poravnanje koristi se Grid, odnosno mreza. Treba se ici ka tome da se je u softveru uvek isto ili slicno poravnanje kako bi se korisnik manje-vise navikao na pozicije odredjenih elemenata. Na primer, ako je to poruka o gresci, onda napraviti tako da je "OK" dugme uvek u donjem desnom uglu. Ta organizacija moze biti unutrasnja ili spoljasnja. Unutrasnja ako je pitanje softvera, a spoljasnja ako platmorfma diktira pozicije elemenata.

Web stranice su vise isle ka tome da otklone manu GUI-a, a to je niska prenosivost. Web kao dizajnerski interakcioni prostor zato ima manje widgeta, ali mu je prenosivost nemerljiva. Zelja je bila da svaki browser mora da podrzi tu grupu widgeta cime je prenosivost manje-vise postala maksimalna. Data grupa je tako napravljena da je pomocu nje moguce napraviti maltene sve. Ako se napravi web aplikacija sa widgetima iz date grupe, svi na ovom svetu ce moci da vide te widgete i da ih koriste u okviru svog pretrazivaca. Medjutim, to nije bas dugo potrajalo. Razlog lezi u tome sto su ljudi poceli sve vise plug-in-ova da stavljaju zarad atraktivnosti, bolje interakcije svoga sajta. Problem sa plug-in-ovima jeste taj da ako ih korisnik ne poseduje, nece moci videti dati widget. Time je znacajno ugrozena prenosivost Web stranica. Home stranica svake web aplikacije mora imati 4 primarna elementa. To su identitet, koji definse gde smo stigli. Potom navigacija da dodjemo do svih grana sajta. Sadrzaj, u smislu slika, cudnog teksta koji ce naterati korisnika da udje u odredjenu stranicu sajta itd. Takodje, neophodno je i

postojanje alata kao sto su search i directory. Web stranice trebaju biti adaptivne u smislu da se trebaju prilagoditi prikazu, pristupanju sa razlicitih uredjaja. Adaptivnost se ne odnosi samo na prilagodjavanje po velicinu, grafickoj moci prikaza, vec i u odnosu na okruzenje u kojem ce ono biti konzumirano.
Ako se recimo pristupa web sajtu iz automobila, onda je neophodno, pored adaptivnosti po velicini, izvrsiti i adaptivnost u zavisnosti od okruzenja, a to je automobil. U tom slucaju obezbediti, recimo, mogucnost search pomocu govora. Neophodno je obezbediti i selektivni prikaz. To znaci da se korisniku, po njegovoj zelji, prikazuje samo onaj sadrzaj koji on zeli. Vazno je obezbediti i tekstualnu alternativu. Na primer, ako postoji video u kojem se nesto prica, nesto objasnjava, onda dati alternativu u vidu teksta za ljude koji mozda ne cuju. Treba se voditi i racuna o tome koju rezoluciju najcesce korisnici koriste, jer sa manjom rezolucijom ide i manji sadrzaj. Vazno je voditi racuna i o browseru koji koristi. Na osnovu poznavanja rezolucije i browsera moze se proracunati "safe area". To je prostor koji imamo na raspolaganju i kada korisnik prvi put otvori nasu stranicu taj "safe area" koji mu se prikaze jeste ono cime ga odmah, prvo napadamo.

Mobilni uredjaji predstavljaju takodje jedan od dizajnerskih prostora. Oni imaju svoj skup widgeta, koji je naravno podrzan njihovim operativnim sistemom, i oni su takvi da su prilagodjeni malim prikazima, niskoj potrosnji baterije itd. Widgeti su skromniji, ali mogu biti potencijalno bogatiji od GUI-a i Web-a sto se tice interakcionih uredjaja. Na primer, ako se pojavi neki proizvodjac softvera, koji je pritom izuzetno cenjen, i napravi neki novi softver za moj telefon, a taj softver zahteva neki novi interakcioni uredjaj, onda cu se ja, kao kreator telefona, potruditi da i ugradim zahtevani interakcioni uredjaj. Na taj nacin se dobio jos jedan interakcioni uredjaj, samim tim i interakciona tehnika kojom ce se moci opet dodatno upravljati i widgetima.

Ugradni sistemi su oni sistemi koji su ugradjeni u odredjenu masinu i postali su deo te masine. Korisnik koristi datu masinu preko datog ugradnom sistema, pa ti sistemi trebaju da obezbede samo taj deo posla koji ta masina radi. Uglavnom su ugradjene u neka brza tehnicka postrojenja pa se zahteva da su izuzetno brzi i sigurni. Takodje, cesto se desava da data postrojenja zahtevaju posebne uredjaje, pa ih dati ugradni sistemi, kojima se masina kontrolise, moraju podrzati. Po tome su ugradni sistemi specificni. Sto se tice widgeta, nalaze se negde izmedju Web-a i GUI-a. Poseban naglasak kod datih sistema jeste na prevenciji greske, alarmima, real-time radu i pracenju standarda u izvodjenu softvera.

Sveprisutno racunarstvo se bazira na tome da nije vise korisnik taj koji pokrece interakciju, nego sistem prati korisnika i kada se zakljuci, iz njegovog ponasanja, da ce mu odredjeni servis biti potreban, taj servis njemu bude i dat. Iz prethodno ispricanog se lako da zakljuciti da vise nema widgeta. Widgeti su eventualno ostali samo na nivou podesavanja sistema. Pitanje je da li se ovakvo racunarstvo moze svrstati u dizajnerski interakcioni prostor, ali i u HCI uopste.

Gordon Mur je jos davne 1965. godine dao predvidjanje po kome ce se broj tranzistora na istoj povrsini procesora/cipa svakih 18 meseci udvostruciti. Dato predvidjanje je jos poznato i kao Murov zakon ("Moore's law"). Procesori su poceli toliko da napreduju da se vec krajem 20. veka doslo u situaciju gde su pocele da pristizu nove generacije procesora, a da prethodne u potpunosti nisu iskoriscene. Iz Murovog zakona je proistekao jedan drugi, Bakstonov. On prica o tome da ako svakih 18 meseci dobijamo sve jace i jace procesore, pa onda ce i masine u koje su dati procesori ugradjeni poceti pruzati sve vecu funkcionalnost. Pored Murovog i Bakstonovog zakona, postoji jos jedan, Boziji. Boziji zakon prica o tome da se covekove mogucnosti kroz godine ne menjaju. Mozda izgledaju malo drugacije, malo su obrazovaniji, pismeniji, ali sto se tice mogucnosti, one su iste. Preklapanjem linije razvoja racunara (Murov i Bakstonov zakon) i linije covekovih mogucnosti (Boziji zakon) dobijaju se dva perioda. Prvi period, do nekih 90 godina, u kome su ljudi bili mocniji od masine, u smislu broja funkcija koje pruzaju i njihovih kompleksnosti. Medjutim, posle 90 godina se preslo u period gde racunari mogu da rade mnogo

vise funkcija od coveka, a pritom su te funkcije izuzetno kompleksne. Za same kreatore interakcionih sistema, mnogo je laksi bio period pre 90-ih godina. Razlog lezi u tome sto su mogucnosti coveka bile mnogo vece u odnosu na masinu i bilo mu je mnogo lakse objasniti sve funkcionalnosti koje ona pruza. Posle 90-ih godina se doslo do toga da je lepeza funkcija, pritom i njihova kompleksnost, sve veca i veca, te ih je sve teze i teze objasniti i predstaviti coveku. Prema tome doslo je do podele u razvoju tehnickih resenje na system centered design (pre 90-ih) i user centered design (posle 90-ih). System centerd design se bazirao na tome da je cilj pravljenje masine/racunara/softvera, a korisnik je nadmocniji te ce lako shvatiti kako se koristi. User cenetered design se zasniva na tome da fokus nije samo na masini, vec i na korisniku. Za razliku od system cented design gde je masina bila u fokusu, kod user centered design korisnik je centar.

User centered design se oslanja/ogleda u sledecim stvarima. Pre svega na inkluzivni razvoj. To znaci da korisnik mora biti ukljucen u sve faze razvoja softvera/hardvera, cega god. Znaci ne samo na pocetku, gde se prikupljaju podaci i zahtevi, i na kraju, prilikom testiranja, vec tokom celog trajanja razvoja. Naravno, prisutnost korisnika je najvaznija na pocetku i kraju, ali je njegova prisutnost obavezna i tokom razvoja da bi se uvek obezbedio taj neki feedback projektantskom/razvojnom timu. Pritom, njegova prisutnost ne treba biti samo pasivna, u smislu da bude izvor informacija, vec i aktivna. To znaci da uvek treba da daje svoje misljenje. Naravno, tu postoji odredjeni problem. Problem lezi u tome sto korisnik najcesce ne zna tehnologiju kojom se pravi taj softver/hardver, pa se postavlja pitanje na koji nacin da dobijamo informacije o nasoj daljoj odluci. Zbog toga je ukljuceno nesto sto se zove prototip. Koriscenje prototipa se zasniva na tome da se napravi neko pojednostavljeno resenje, iz kojeg se izbaci sve sto trenutno nije od interesa. To pojednostavljeno resenje se potom moze dati korisnicima na koriscenje, te se moze cuti i njihovo misljenje. User centered design jeste prva pragma koja je ukljucila prototip, kao mogucnost aktivnog ucesnja korisnika.

Stakeholders jeste termin koji je user center design ubacio u softversku praksu. User centered design je prosirio pojam korisnika i ubacio stakeholderse. Stakeholdersi jesu osobe koje rade sa datim softverom, odnosno imaju bilo kakav kontakt/interes sa njim. Prema tome, ne postoje korisnici, vec stakeholdersi. To znaci da prilikom razvoja softvera, on se razvija za sve stakeholderse, za sve grupe ljudi, za celo drustvo, koje ce imati interes od datog softvera. Prema tome, drustvo je podaljeno u tri grupe. Grupe su primarni, sekundarni i tercijarni stakeholdersi. Primarni stakeholdersi su oni korisnici koji odgovaraju korisnicima iz system centered design-a. To su bukvalno oni korisnici koji direktno interaktuju sa softverom, klikcu po ekranu itd. Medjutim, user centered design je sada ubacio stav: Nemoj kada pravis interfejs da na umu imas samo ovu grupu korisnika, vec imaju i ostale korisnike. Sekundarni korisnici, odnosno stakeholdersi, su oni korisnici koji nisu u direktnoj interakciji sa softverom, ali preko odredjenih outputa/izlaza tog softvera oni stupaju u interakciju sa njim. Ako se uzme primer elektrovojvodine, primarni stakeholderi jesu operateri koji sede za racunarima i "pisu racune", a mi koji ih dobijamo smo sekundarni. Taj racun je dosao kao rezultat rada tog softvera nad kojim su radili primarni stakeholderi. Prema tome, treba se uzeti u obzir i izgled tog izlaza. Medjutim, postoji i treca grupa. Tercijarni stakeholderi su svi oni ostali ljudi koji imaju neke interese od datog softvera. U grupu moze spadati direktor elektrovojvodine (ukupan obracun), kolege koje odrzavaju softver (za njih neka dokumenta o radu softvera) itd. Stakeholdersi i inkluzivni razvoj, kroz koriscenje prototipa, su bili osnovi pocetnog user centered designa. Sa pojavom mobilnih uredjaja i shvatanje user centered designa se prosirilo, jer se sa njihovom pojavom razvoj softvera promenio. Prema tome, po user centered design, prilikom razvoja softvera, se mora voditi racuna o tome ko ce koristiti softver (stakeholdersi), koje ce zadatke obavljati sa njima, ali uvedena je i treca stvar, to je kontekst upotrebe. Kontekst upotrebe se bazira na tome da ce se softver prilagodjavati situaciji u kojoj stakeholder rukuje sa njim. Ako radi u kancelariji, onda je to desktop resenje, ako je mobilni telefon, onda je mobilno resenje.

Projektovanje intreakcionih sistema se sastoji od nekoliko faza. Prva faza jeste formiranje zahteva. Foramiranje zahteva se sastoji od kreiranja toga sta mi zelimo da uradimo. Sve ostale faze u razvoju softvera se oslanjaju na fazu formiranja zahteva. Greske koje nastanu ovde se posle samo povecavaju i povecavaju i sve ih je teze ispraviti. Da bi se na pravi nacin formirali zahteva neophodno je razumeti stakeholderse, njihove zahteve, ali i kontekst upotrebe. U ovoj fazi otkalanjanje greske najmanje kosta. Formiranje zahteva je najbolje obaviti na sto vise nacina. Ono se u stvari bazira na jednoj petlji, jednom krugu tri aktivnosti. Te aktivnosti u stvari rade zajedno da bi se na kraju dobio jedan kompetan skup zahteva. Prva aktivnost jeste sakupljanje podataka o stakeholdersima, zadacima i kontekstu upotrebe. Potom analiza i interpretracija podataka. Posto prilikom sakupljanja podataka dodje do priliva informacija, neophodno ih je interpretirati. Kroz interpretaciju se vidi koji su to podaci isti, odnosno identicni. Kada se to odradi, prelazi se u analizu. Nakon analize, prelazi se u odredjivanje zahteva.

Nekoliko je kljucnih stvari kada je sakupljanje podataka u pitanju. Prvo treba da se definise cilj sakupljanja podataka, jer ako se bez cilja ide u sakupljanje, doci ce se do toga da na analizu, kao druge fazu formiranja zahteva, stigne jedan ogroman talas podataka od kojih mnogi nece imati velikog znacaja za nas. Treba da se ostvari i dobar odnos sa korisnicima. Samo na taj nacin mogu da se izvuku prave informacije iz njih. Takodje, vazno je i trianglacija. Trianglacija se odnosi na to da se koriste razlicite tehnike prikupljanja nad razlicitim korisnicima i u razlicito vreme. Moze se desiti da kod odredjenih korisnika mi, kao sakupljaci podataka, ne prodjemo bas najbolje. Zato je nekad vazno sprovesti pilot studije koje ce omoguciti korisnici da vidi na koji nacin ce se vrsiti prikupljanje podataka. Pilot studije su dobre i za nas kako bi nam dale uvid da li ce data tehnika prikupljanja podataka za te ljude u datom trenutku proci na nacin koji smo planirali. Postoje razliciti nacini na koje se mogu beleziti podaci. Najbolje resenje jeste njihovo kombinovanje. Nacini su sledeci: pisanjem, audio, video i kompjuterski loging. Pisanje je najstarija tehnika, pritom je i izuzetno jeftina, ali je veoma zamorna, kako za pisanje, tako i za kasnije citanje. Audio i video cesto ide u paru i predstavlja izuzetno dobro resenje, ali se nedostatak ogleda u tome sto pojedine situacije koje se jednostavnu u tom trenutku iz nekog razloga nisu desile ce izostati, a i korisnici mogu da pocnu da glume. Kod pisanog nacina belezenja podataka, tog izostavljanja nema. Kompjuterski loging se bazira na tome da kompjuter automatski prikuplja podatke tokom kretanja korisnika kroz softver. Taj softver moze biti prototip ili starija verzija koja ce nam sluziti kao izvor podataka za razvoj nove. Podaci se mogu prikupiti na razlicite nacine, odnosno putem razlicitih tehnika. Te tehnike ce biti predstavljene u nastavku.

Prva tehnika jeste prikupljanje podataka putem anketa. Ankete su skup pitanja i ponudjenih odgovoru. Na osnovu izbora odgovora od strane korisnika, mi kao sakupljaci podataka donosimo odredjene zakljucke. Najbolja pitanja su ona koja kao ponudjene odgovore imaju samo da i ne. Korisnik takva pitanja vrlo brzo shvata i jos brze odgovara na njih. Takodje, obrada tako definisanih anketa ide veoma brzo. Medjutim, prethodno opisane ankete su veoma retke. Cesto se desava da pitanja imaju vise odgovora. Tada se mora voditi racuna o tome da pitanja u okviru sebe nemaju vise potpitanja, kao i to da odgovori budu jasno definisa i medjusobno razliciti. Ne sme se korisnik dovesti u situaciju da vaga izmedju dva odgovora da osnovu njihovog opisa. Ovakve zatvorene ankete, odnosno one u kojima postoje ponudjeni odgovori, daju konkretne podatke, tacnije ne postoji nista izmedju. Cesto se desava da su pitanja takva da nije moguce jasno definisani odgovore koji ce biti ponudjeni. U datim situacijama se korisniku ostavlja mogucnost da na linijama, ostavljenim ispod pitanja, napise odgovor. To su takozvana otvorena pitanja. Ona su dobra jer korisnik tacno moze da napise svoje misljenje o nekoj odredjenoj stvari. Nevolja se ogleda u tome sto tada analiza postaje izuzetno teska. Teska u smislu da treba procitati sve te odgovore od svih tih ljudi, a i cesto se desava da su odgovori negde izmedju. Cesto se u anketama postavljaju i kontrolna pitanja. To su pitanja koja pitaju vec pitano, ali na drugi nacin i sa drugim odgovorima. Time se proverava iskrenost korisnika. Kada se utvrdi da na vise kontrolnih pitanja dolazi do

nepoklapanja, takav anketni listic se odbacuje. Treba se istaci i to da je veoma vazno za validnost ankete da ona bude na dobrovoljnoj bazi. Na kraju, treba istaci da postoji dve vrste anketa, papirna i elektronska. Danas su cesce elektronske. Prednost elektronskih se ogleda u tome sto su jeftinije, ali i brze, jer se automatski vrsi analiza prikupljenih podataka. Papirne takodje imaju neke prednosti. Prednost se ogleda u sigurnoj anonimnosti. Cesto se postavlja pitanje kada je to dobro koristiti anketu kao sredstvo za prikupljanje podataka. Dobro je onda kada znamo na koji nacin da postavimo pitanja i odgovore. To mozemo znati samo ako smo dobro upoznati sa oblasti za koju je data anketa vezana. Druga stvar koja utice na izbor jeste broj ljudi od kojih je potrebno prikupiti podatke. Ako je taj broj mali, onda je pitanje da li je treba sprovesti. Razlog lezi u tome sto onda postoje mnogo bolje tehnike. Medjutim, ako je broj ispitivanih osoba veliki, onda je anketa izuzetno dobra.

Druga tehnika za prikupljanje podataka jeste intervju. Ona se bazira na tome da osoba koja prikuplja podatke, kao i osoba koja ih daje, najcesce je to buduci stakeholder, sednu i pricaju. Izuzetno je efikasna tehnika jer ljudi mnogo vise vole da pricaju, nego da popunjavaju "dosadne" ankete. Intervju moze da se ostvari na razlicite nacine, licem u lice, telefonom, preko racunara itd. Najproduktivnije jeste upravo uzivo, licem u lice. Data tehnika nije dobra kada je potrebno ispitati veliki broj ljudi. Jednostavno nije moguce pricati sa tolikim brojem buducih stakeholdersa, pogotovo ako vreme igra vaznu ulogu. Intervju se sastoji od nekoliko delova. Prvi jeste "introduction". On sluzi da pripremi sagovornika, odnosno da pripremi teren. Nakon toga ide "main body". U datom delu se realizuje proces postavljanja pitanja i belezenja odgovora. Najcesce se to belezenje odgovora ostvaruje putem audia ili videa. Ta pitanja mogu biti nestruktuirana, struktuirana i semistrukturiana. Nestruktuirana pitanja su ona koja nastaju u hodu, tacnije ona koja sama proisteknu tokom price sa sagovornikom. Jednostavno receno, nisu ranije pripremljena. Struktuirana su ona koja su ranije pripremljena. Problem sa struktuiranim jeste taj sto ona nekada nisu dobro pripremljena, pa je potrebno ponovo odraditi intervju. Semistruktuirana jesu kombinacija prethodna dva. Najbolji nacin prikupljanja podataka, ali i najskuplji sto se tice vremena. Intervjuo ne trosimo samo nase vreme, vec i vreme osobe koja nam je sagovornik.

Veliko trosenje vremena je reseno putem intervjua grupe. Bazira se na tome da sada ne postoji samo jedan sagovornik od kojeg dobijamo podatke, vec citava grupa ljudi. Kada je intervju grupe u pitanju onda najcesce ne postoje neka struktuirana pitanja, vec se sve bazira na tome da se jednostavno postavi neko pitanje koje ce pogurati ostale sagovornike da medjusobno komuniciraju i raspravljauju. Sakupljaci podataka u datoj situaciji trebaju da sluze samo kao osobe koje ce zapisivati podatke dobijene iz tih razgovora, ali i kao koordinatori. Kordinatori u smislu da ukljuce svakog u razgovor a i prekinu mozda ljude koji "previse pricaju". Dobra stvar grupnog intervjua jeste taj sto vise podataka od vise ljudi dobijamo za krace vreme, kao i to sto sagovornici medjusobno cuju jedni druge pa mogu da se nadopunjuju. Na taj nacin se dobija jedna mnoga kompletnija slika nego kada je intervju sa jednom osobom u pitanju. Nedostatak je taj sto su to ipak ljudi, pa kada je veca grupa u pitanju mozda nece biti bas u potpunosti iskreni i otvoreni. Zato te grupe trebaju biti sacinjene maksimalno od 10 clanova. Medjutim, ni to nije garancija otvorenosti i iskrenosti sagovornika.

Sledece tehnike su direktna ili indirektna opservacija. Direktna opservacija se bazira na tome da se ode na radno mesto i direktno posmatraju osobe od interesa, dok se indirektna zasniva na tome da se ne odlazi na radno mesto vec se putem audio i/ili video uredjaja vrsi posmatranje. Kod direktne opservacije prednost lezi u sto sam ja odmah na radnom mestu te istog trenutka kada mi nesto nije jasno mogu da pitam, trazim obrazlozenje. Nedostatak je taj sto sam ja, kao sakupljac podataka, ipak tamo prisutan, te mogu da izazovem nelagodnost, ali i da potencijalno smetam u obavljanju delatnosti. Takodje, moze to biti firma koja radi 24/7, pa ne mogu ja biti prisutan svo to vreme. To se moze resiti vecim brojem sakupljaca podataka, ali to onda nisu iste oci, nije isti ugao posmatranja. Za date situacije je bolja indirektna opservacija. Medjutim, indirektna opservacija ima mane koje se ogledaju u tome da

onda ne mogu biti postavljena pitanja radi dodatnog objasnjenja, ali i ugao kamere moze biti takav da se nesto ne vidi, jednostavno propusti. Mozda i najveca mana lezi u tome sto onda neko mora te silne snimke posle gledati. Mana i jedne i druge lezi i u tome sto su to ipak ljudi, te je gluma uvek prisutna.

Naredne tehnike cak ni ne ukljucuju buduce stakeholderse naseg softvera. Prve jeste proucavanje dokumentacije. Na osnovu dokumentacije, koja se moze citati kod kuce, u firmi, u kaficu, bilo gde, moze se na jedan lep nacin shvatiti kako data firma/preduzece funkcionise. Naravno, da bi to bilo efikasno, dokumentacija mora biti napisana na odgovarajuci nacin. Data tehnika je jako dobra ako prethodno nismo upoznati sa delatnoscu kojom se ta firma bavi. Prvo se putem dokumentacije upoznamo sa firmom, shvatimo na koji nacin funkcionise, a posle se pristupa nekim drugim tehnikama, kao sto je recimo intervju, kako bi se dobili jos detaljniji podaci. Najveci problem date tehnike lezi u tome sto tu obicno ima jako pune dokumentacije, a i moze se desiti da opet ne shvatimo na pravi nacin funkcionisanje preduzeca. Takodje, ne zahteva oduzimanje vremena stakeholdersima. Druga tehnika koja ne ukljucuje stakeholderse jeste proucavanje slicnih proizvoda. Jednostavno na osnovu vec postojacih proizvoda iste namene mozemo sakupiti dosta korisnih podataka.

Nakon sto se odradi sakupljanje podataka, prelazi se u proces interpretacije istih. U procesu interpretacije se gleda da li odredjeni podaci imaju isto znacenje, samo su na drugaciji nacin iskazani. Interpretacija sluzi kao neki filter pre prelaska u proces analize prikupljanih podataka.

Posle uspesne interpretacije se prelazi u obradu dobijenih podataka, odnosno u proces definisanja toga sta korisnici zele. Postoje dve vrste analize, kvantitativna i kvalitatitvna. Kvantitativna analiza predstavlja dobijanje brojeva, vrednosti, procenata, frekvencije pojavljivanja na osnovu obrade podataka. Moze se reci da kvantitativna analiza predstavlja statisticku obradu. Kvalitativna analiza se bazira na tome da se iz mnostva grubih podataka izvuku najvaznije informacije, odnosno podaci od interesa, odredjeni transkripti.

Uspesna analiza nas odvodi i do poslednje faze pri kreiranju zahteva, a to je odredjivanje, definisanje zahteva. Postoji nekoliko vrsta zahteva koji trebaju biti definisati. Prvi su arhitektronski zahtevi. Predstavljaju zahteve koji proisticu iz arhitekture sistema za koji pravimo softver. Na primer, ako se pravi softver za neku banku, onda taj softver treba da podrzi i da se pridrzava svih bezbednosti mehanizama koji proizilaze iz arhitekture date banke. Sledeci su funkcionalni zahtevi. To su svi oni koji proisticu iz definisanja zadataka koje ce dati softver obavljati. Potom imamo i nefunkcionalne zahteve. U njih spadaju svi ostali zahtevi. Na kraju, treba spomenuti i zahteve ogranicenja. U njih spadaju sva ona ogranicenja koja su jako bitna. Posebno ako se radi o softveru koji je jako opasan, kritican.

Spoznavanje korisnika prilikom razvoja softvera je veoma bitno, jer ako ne znamo korisnike nije moguce ni napraviti softver maksimalne upotrebljivosti. Pored onih osnovnih karakteristika korisnika, kao sto su starost, obrazovanje, kulturno nasledje itd., potrebno je spoznati i odredjene specificne karakteristike. Te specificne karakteristike mogu biti daltonizam, autizam, invladitet itd. Da bismo spoznali sve korisnike i dobili podatke o njima mogu se koristiti neke od prethodno opisanih metoda za sakupljanje podataka. Medjutim, sta raditi ako nas softver treba da bude globalan. U pomoc su nam pristigli psiholozi. Oni su nam omogucili, odnosno objasnili prototipiziranje korisnika. Prototipiziranje korisnika se bazira na tome da se prave prototipovi korisnika, poznatiji kao personas. Personas, kao prototipovi buducih stakeholdersa, su promovisani od Alana Kupera. Personasi su karakteristicni predstavnici vecih grupa korisnika, gde se onda kroz te predstavnike gledaju zahtevi korisnika, odnosno spoznaju korisnici na pravi nacin. Softver ce biti dobar onoliko koliko su personasi tacni. Drugi nacin sa spoznavanje korisnika jeste kada sebe stavimo u "njihovu kozu". Dati nacin se koristi kada ni sami nismo sigurni ka cemu idemo, odnosno kada zelimo da napravimo nesto novo. Primer jeste razvoj mobilnih

uredjaja od strane Dzef Havkinsa. On se zalagao za razvoj malih uredjaja koji nece pruzati mnogo, ali ce obezbediti bas ono sto treba.

Kada smo upoznali korisnike, onda trebamo da upoznamo i njihove zadatke, odnosno sta oni to hoce da rade sa softverom. Prvo treba definisati cilj/ciljeve, tacnije sta on hoce da postigne prilikom rada sa softverom. Potom koje zadatke treba da obavi da bi ispunio cilj, gde svaki od tih zadataka ima odredjene akcije. Implementacija softvera se bazira na preslikavanje datih akcija u funkcije softvera. Pozivanjem akcija se ostvaruju razliciti zadaci, a na kraju i sam cilj. Problem je kako doci do tih zadataka, kako ih opisati. Za to postoje razclicite tehnike. Postoje dve grupe notacija, tehnika za zapisivanje zadataka. Jedna grupa notacija, tehnika za zapisivanje zadataka koje nas softver treba da ispuni jeste How to Do it. Ta grupa tehnika opisuje samo sta treba da se radi, sta treba da se obezbedi korisniku da uradi sa nasim softverom. Druga grupa tehnika jeste kognitivna analiza koja pored How to Do it uvlaci i sta se korisniku desava u glavi da bi taj How to Do it izasao sa rezultatom.

Kada smo upoznali korisnike, onda trebamo da upoznamo i njihove zadatke, odnosno sta oni to hoce da rade sa softverom. Prvo treba definisati cilj/ciljeve, tacnije sta on hoce da postigne prilikom rada sa softverom. Nakon toga, treba definisati zadatke koje treba obaviti da bi se ispunio neki cilj. Ti zadaci mogu da se obavljaju u razlicitim sekvencama ili paralelama, razlicitim putanjama, gde se do svakog cilja moze stici razlicitm putanjama. Treba se voditi i racuna o tome da pojedini zadaci mogu biti izuzetno slicni (pojavljivanje istih ili slicnih zadataka) u okviru razlicitih ciljeva. Jednom recju, dati zadaci su frekventni. U datim slucajevima treba to napraviti kao jedan dijalog koji ce se pojavljivati prilikom ostvarivanja razlicitih zadataka. Prilikom kreiranja interfejsa, data frekventnost odredjenih zadataka ili akcija je veoma vazno zbog postavljanja na odgovarajuce mesto u okviru interfejsa. Niju je potrebno ponovo praviti! Takodje, treba se voditi i racuna o tome da li je dati zadatak kritican, da li zahteva samostalan ili grupni rad itd. Svaki od tih zadatak treba podeliti u odredjene akcije. Implementacija softvera se bazira na preslikavanje datih akcija u funkcije softvera. Pozivanjem akcija se ostvaruju razliciti zadaci, a na kraju i sam cilj. Problem je kako doci do tih zadataka, kako ih opisati. Za to postoje razclicite tehnike. Postoje dve grupe notacija, tehnika za zapisivanje, predstavljanje zadataka. Jedna grupa notacija, tehnika za zapisivanje zadataka koje nas softver treba da ispuni jeste How to Do it. Tehnike koje pripadaju datoj grupi opisuju samo koje akcije postoje u okviru koji zadacih i ka kojim ciljevima vode. Druga grupa tehnika se zasniva na kognitivnoj analizi koja pored How to Do it uvlaci i sta se korisniku desava u glavi, sta mora da zna, koje vestine ima da bi taj How to Do it izasao sa rezultatom. Data grupa tehnika je izuzetno vazna za HCI, jer objasnjava ponasanje korisnika. U nastavku ce biti objasnjeno po 3 tehnike iz svake grupe.

Prva tehnika u okviru grupe tehnika "How to do it" jeste scenariji. Tehnika je preuzeta iz kolaga sa dramaturgije. Bazira se na tome da nam korisnik napise ili isprica na koji nacin nesto radi ili na koji nacin zeli nesto da uradi. Scenarij je dobar jer svako moze da ga napravi i svako moze da ga razume. Problem kod scenarij jeste taj sto se prica i pise zivm jezikom. Ziv jezik je po prilicno dvosmislen tako da mnoge stvari mogu biti ili izostavljene ili definisane na razlicite nacine. Cesto se desava da covek prilikom te price ili pisanja preskoci akcije koje se ne desavaju svaki dan.

Sledeca tehnika u okviru iste grupe jeste koriscenje use case-ova. Bazira se na tome da se definise jedna tabele gde se u jednom redu navodi sta radi korisnik, a u redu pored sta radi sistem kao odgovor na tu korisnikovu akciju. Vrlo je dobra tehnika za definisanje, belezenje zadataka jer omogucava laku podelu posle izmedju kreatora interfejsa i kreatora softvera.

Naredna tehnika je unapredjenje use case-ova i pravljenje essential use case-ova. U datoj tehnici se takodje korise use case-ovi, ali se oni oslobadjaju svih tehnoloskih zavisnosti. To omogucava da se dati

use case-ovi koriste i za novi sistem, jer je prilikom njihovog kreiranja izbaceno sve sto je, sa stanovista tehnologije, imao onaj stari. Nije vazno kako obavlja, vec sta obavlja. Tehnologija se nekad i ovde ubacuju ako je vazna radi interakcije sa drugim sistemima koji se nalaze u okruzenju. Nacin na koji ce se nesto obavljati ostavlja se za fazu implementacije.

Svi prethodni metodi su desinisali, opisivali samo sta treba da se radi oslanjajuci se na covekovu stranu i stranu masine. Medjutim, tehnike koje se baziraju na kognitivnoj analizi, pored prethodno recenog objasnjavaju i to sta se desava u covekoj glavi, na koji nacin od razmislja prilikom rada sa softverom. Jedna od takvih tehnika jeste i eXtended User Action Notation (XUAN). Prvo je bio samo UAN, a kasnije je dodato i to X. Slicno je use case-vima. Opet postoje dve strane, dve kolone, ali se svaka od njih opet deli na dva dela. Ta dva dodatna dela su vidljiva i nevidljiva akcija, odnosno vidljiva i unutrasnja akcija. Vidljive akcije korisnika su one koje se vide (klik, pomeranje ruke itd.), a unutrasnje akcije su one prave kognitivne, tacnije sta se desava u covekovoj glavi. S druge strane i kompjuter ima dve kolone. Opet vidljive i nevidljive akcije. Vidljive su one koje se prikazuju na ekranu ("feedback"), dok su nevidljive one koje se desavaju u kodu softvera. Posle je ovo prvobitno shvatanje, tacnije ovaj UAN prosiren sa X (extended) koje oznacava stanje interfejsa. Stanje interfejsa sluzi da oslika stanje sistema koje je izazvano odredjenom akcijom korisnika. Definisanjem stanja interfejsa se dolazi do toga da se onemoguci, presretne mogucnost pojave neke nekonzistentnosti. Time se omogucava i to da se definise na koji nacin ce widgeti da se ponasaju za odredjeno stanje sistema.

Dijagrami toka interfejsa jesu sledeca tehnika za predstavljanje, definisanje, zapisivanje zadataka korisnika. Dijagrami su procvetali sa pojavom mobilnih telefona i bankomata. U datom dijagramu postoje stanja, koja oznacaju stanje interfejsa. Pod stanjem interfejsa se misli na to sta ima na interfejsa i u kom obliku, misli se na izgled interfejsa. Naravno putem dijagrama su definisani i ti prelazi iz jednog stanja interfejsa u drugo, koji proisticu kao rezultat delovanja korisnika. U okviru dijagrama je definisano i to sta treba da korisnik uradi kako bi doslo do prelaska iz jednog stanja u drugo. Pogotovo se probio sa pojavom bankomata i velikim brojem ostavljenih kartica. Dijagram toka interfejsa je veoma koristan jel se moze vrlo brzo utvrditi na koji nacin covek razmislja prilikom interakcije sa softverom.

Poslednja tehnika nam dolazi iz Italije. U pitanju je ConcurTaskTrees. Sluzio je kao osnova za softver koji vrsi zemaljsku kontrolu leta milanskog aeorodroma. Bazira se na tome da se zadaci predstave u obliku jednog stabla gde je moguce njihovo konkuretno izvrsavanje.

Kada je u pitanju balansiranje izmedju automatizacije i korisnikove kontrole treba istaci postojanje dva trougla. Na pocetku je covek bio taj koji je sve radio, dok je masina samo malo pripomagala ili cak nista nije radila, jer nije ni postojala. S prolaskom vremena, covek je razvijao sve bolje i jace masine, tako je on poceo sve manje da radi, a masina sve vise umesto njega. Sada se postavlja pitanje da li treba da mi ne radimo nista, a masine sve? Odgovor je jednostavan i on glasi ne. Treba se doci do toga da masine rade veliki deo posla, ali da nikad se ne predje ta granica da masine rade sve, a mi nista. Razlog lezi u tome sto covek treba da bude taj koji ce dati dozvolu masini da uradi nesto. To moze otici toliko daleko da masina bukvalno sve odradi, a covek samo pritisne dugme da to potvrdi. Medjutim, sama ta potvrda je vazna jer se time sve stavlja pod covekovu kontrolu. Naravno, ni ovde se ne treba preterivati. Odredjene akcije koje nisu "kriticne" i koje bi svakako covek potvrdio treba ostaviti na masini i da pripremi i da pokrene.

Tokom razvoja softvera je vrlo vazno koriscenje prototipova, kako bi upravo tokom njegovog razvoja dobili misljenje korisnika od razvijanom softveru. U razvoju sofvera, sto se tice HCI, postoje 3 vrste prototipova.

Prvi su low fidality prototipovi. To su prototipovi koji su vrlo niske verodostojnosti. Ne lice u mnogome na ono kako ce izgledati sistem. Primenjuju se u prvim fazama razvoja softvera. Sluze samo kao nagovestaj korisniku kako ce otprilike izgledati softver. Insistira se na sto manje detalja i ono sto se predstavi preko datog prototipa jesu samo one glavne stvari koje ce dati interfejsa pruzati. Davanje low fidality prototipova korisnicima i njihove primedbe resavaju zaista one stvari koje su najopasnije, koje su najocigledije. To su uglavnom izuzetno objektivne primedbe. Ne bi se dobro zavrsilo ukoliko kod high fidelity prototipova se potkrade neka objektivna primedba. Na primer, da smo zaboraviti neki citav zadatak da se uradi.

Sledeci su medium fidelity prototipovi. To su u stvari low fidality prototipovi u koje su ukljuceni vise dizajnerskih elemenata. Vec se pocinje pricati o velicini fonta, bojama, stilu itd. Medjutim, pocinje se pricati i o toku interakcije. Dok su low fidality prototipovi bili ograniceni na "look", odnosno na to kako ce izgledati, medium unose i neko njegovo ponasanje. Ovde vec pocinju subjektivne primedbe korisnika.

Poslednji prototipovi jesu high fidelity prototipovi. Veoma se verodostoji. Veoma nalikuju krajnjem proizvodu. To znaci da oni iza sebe moraju imati i funkciju. Razlika izmedju hight fidelity prototipova jesu male sitnice. Ovde je najveci procenat subjektivnih primedbi.

Pomocu cula covek prihvata informacije iz okoline, potom ih obradjuje i reaguje na to sto je primio. Na nama je da proucimo cula coveka i iskoristimo to znanje u cilju pravljenja softver koji ce plasirati informacije na onaj nacin na koji mi zelimo da ih covek primi. Primarno culo za prihvat i obradu informacija jeste culo vida. Sledeca vazno culo jeste culo sluha koje nas upozorava i cuva. Takodje, treba spomenuti i culo dodira koje je vazno jer omogucava direktnu interakciju sa racunarima. Ostala cula koje covek poseduje nisu bitna sa stanovista HCI kao naucne oblasti.

Oko radi na sledeci nacin. Kroz socivo prolazi kontrolisana svetlost i pada na mreznjacu. Funkciju kontrolisanja prolaska svetlosti obavlja zenica. Slika je centrirana na poseban deo mreznjace koji se je jedan vid udubljenja i naziva se zuta mrlja. Sastav mreznjace cine cepici i cilindri/stapici. I cepici i cilindri predstavljaju odredjenu vrstu nervnih zavrsetaka. Prema tome, postoje dve klase nervnih zavrsetaka samim tim i dve uloge. Stapici, odnosno cilindri, su zaduzeni da primaju svetlo i reaguje na kolicinu svetlosti koju su primili. Za razliku od njih, cepici, kojih ima tri vrste (oni koji reaguju na crvenu, zelenu i plavu svetlost), sluze da definisu boju koja u sebi ima onoliko svetlosti koliko kazu stapici/cilindri. Ti nervni zavrseci su tako isprepletani i uvrcu se ka sredistu oka i kreiraju opticki nerv. U sredini tog optickog nerva se nalazi slepa mrlja. Onaj deo slike koji pada na slepu mrlju mi ne vidimo. Razlog lezi u tome jer je vec tu kreiran zivac i nema nerava koji bi primili svetlost ili boju. Oko zute mrlje ime vise cepica, a manje cilindara. Odaljavanjem dolazi do sve manjeg broja cepica i sve veceg broja stapica. Najbolje boju vidimo oko zute mrlje, a sto vise idemo ka periferiji tada smo manje osetilji na boje, a mnogo vise na svetlo. Covek je 20 puta osetljiviji na svetlost nego na boju. To znaci da ako se prezentuju neke slike i ako se zeli postici da on uhvati svaki detalj na toj slici onda je bolje da ta slika bude crno bela. Razlog lezi u tome sto su boje ono sto privlaci paznju pa ce biti uvek za tu namenu iskorisceni cepici kojih ima manje i koji nece moci da "renderuju" na pravi nacin te sitne razlike u boji. Ako sliku zasnivamo samo na stetlosti, tada ce se ukljuciti stapici, kojih ima vise, i koji ce to moci da obrade na pravi nacin. Kada je u pitanju broj cepica i broj stapica i nacin na koji covek rezunuje boje i svetlost, to je HCI struka na vrlo dobar nacin iskoristila. Sve detalje koje je potrebno procitati, svaki broj, procenat, znak ce biti smestani u centar vidnog polja. Sve ostalo sto nije bitno ne bi bilo dobro da se izbace u centar vidnog polja kao neka poruka, jer ces prekinuti dok radimo nesto sa softverom. Takve stvari, kao sto je recimo obavestenje, najbolje prikazati u rub vidnog polja i obicno se prikazuje nekom malom animacijom. Svrha animacije je u tome da se pomocu nje izvrsi promena svetla. Samim tim se garantuje da ce korisnik to primetiti, ali ako mu je dalje potrebna paznja za onim sto je prethodno radio, to mu nece smetati. Imamo vise primaoca

(stapica) crvenog svetla, nego zelenog i plavog. U tome lezi razlog sto sto veoma osteljivi na crvenu. Ako radimo za decu, tu treba gurnuti sto vise crvene boje, jer ce oni to sigurno videti. Covek gleda **centricnost**, njegov vid je **centrican**. Za nas progremere je vazno da razumemo nesto sto se zove "eye path". Ono kaze da covek trza oko i hvala detalje, pravi slike. Nakon sto to odradi, onda se koncetrisemo samo na promene. Radi provere centricnosti covekovog gledanja, sproveli su test nad ljudima. Taj test se bazirao na tome da su oni gledali Nefretiti (egipatska carica), a naucnici su putem brzih kamera vrsili pracenje pokreta oka. Eye path, pokretanje oka, se bazira na kretanju od centra, potom hvatanju velikih slika, onda velike naslove i neke elemente koji su mu pokupili paznju. Napravljene su citave nauke koje se baziraju na eye path-u koje omogucavaju da proizvodjaci interfejsa mogu da navedu korisnika tamo gde treba i nacin koji oni zele. Ne treba se mnogo oslanjati na boju zbog daltonista. Sa koje udaljenosti treba gledati prezentacije? Ako su 3D onda sto veci ekrani i sto blize sto nam omogucava da ne vidimo granice i da se stvori stvarno utisak kao da je 3D. Medjutim, da bi gledali blizu tehnologija ne sme da steti, takodje i broj piksela mora biti veci. Sto je veca rezolucija potrebno je biti sto dalje. Kako covek gleda 3D? Svako oko ima svoju sliku, gde ih onda mozak prima i od njih pravi 1 slika koju stavlja ispred sebe. Postoje razlicite 3D tehnologije.

Culo sluha nas cuva. Omogucava nam da cujemo zvuk ispred nas, tada slusamo pomocu usiju, ali postoji mogucnost i da cujemo iza nas, onda slusamo preko usnih kostiju. Usne skoljke su napravljene tako da prikupljaju zvucne signale koji dolaze s polja. A sta mi to slucamo? Slusm zvuk. Zvuk je vibracija/promena pritiska medija u kojem se glava nalazi. Na koji nacin uho radi? Ono je napravljeno tako da primi promenu pritiska vazduha, usmerava ga u usni kanal. Usni kanal ce preneti te promene pritiska na bubnu opnu. Ona ce poceti da se pomera prema unutra ako je veci pritisak nego kod nas u glavi, odnosno prema napolje ako je nizi pritisak nego u nasem unutrasnjem uhu. Cekic, nakovanj i uzengija ce prenositi taj pokret koji napravi bubna opna i pocevati ga cak 33 puta. Tako da ce puz, do kojeg prethodna tri prenesu vibracije i u kome se nalazi jedna gusta tecnost i u okviru kojeg se nalaze dlacice (nervni zavrseci) kako nakovanj pumpa pritisak, vodica pocinje da luduje, mrda nervne zavrsetke, i to je nadrazaj koji mi cujemo kao zvuk. Majka priroda nas je sacuvala od velikih zvukova na sledeci nacin. Taj nacin se bazira na tome da se oko cekica, nakovanja i uzengije nalaze misici koji se pritezu pri vecim jacinama zvuka. Tada pojacanje nije 33 puta. Na taj nacin cuvaju unutrasnje uho od prevelikog pritiska. Mozak pritom dobija informaciju koliko su se oni pritegli, i onda dobijamo osecaj koliko je zvuk jak. Ako se bilo duze u nekom prostoru gde je zvuk glasan, ti misici mogu ostati zategluti. Oni ce postepeno poceti da se opustaju. Ako mi nase misice stalno izlazemo glasnom zvuku, oni ce zaboraviti da se opuste. Ako ostanu zgrceni, oni vise nece vise pocevati 33 puta. Posledica toga jeste gluvoca. Promena pritiska, ako je promena pritiska velika, to zovemo aplitudom, mi to tumacimo kao jacina zvuka. Brze ili sporije promene zvuka zovemo visina. Ako je visa frekvencija, kazemo da je zvuk visok, ako je niza onda je dublji. Na tumacenje zvuka, pored visine i jacine, utice jos nesto. To su harmonici. Predstavljaju boje zvuka, odnosno umnoske osnovne frekvencije. To su razlicite grbe, lakovi itd. Flecer-Munsen psiho-akusticne krive su izuzetno bitne za nas kao kreatore intreakcionih racunarskih sistema. One su nastale tako sto su ljudi stavljeni u gluvu sobu. Soba koja je tako dizajnirana da nece davati harmonike zvuku. To znaci da ce se u sobi cuti samo ono sto emituje zvucnik. U jednom trenutku im puste zvuk jedne frekvencije i jednom jacinom. Potom prestanu emitovati taj signal i kazu im sada cete imati jedan za drugim po 10 razlicitih frekvencija, svaka od njih ce biti prvo tiha pa ce se polako pojacavati. Vas posao je da pritisnete taster kada vam se ucini da je dati zvuk iste jacine kao onaj prvi koji ste culi. Test je uveo revoluciju u shvatanju na koji nacin covek slusa. Test je pokazao da covek najbolje cuje na frekvenciji od 1 do 4 kHz. Razlog lezi u tome sto pricamo tim frekvencijama. Dok pricamo, generisemo zvukove te visine, te smo namestili da nam je slusni aparat najosetljiviji na te frekvencije. Jedinica za zvuk jeste decibal. Covekovo uho je selektivno. Smatra se da od ukupnog zvuka do covekovog uha dodje samo 25% ukupnih informacija. Sav nas mehanizam slusanja prouzrokuje je da se zvuk koji slusamo podeli u 25 podopsega gde su opsezi na nizim frekvencijama jako mali, par stotina Hz, dok je kod visih frekvencija ta razlika i do

nekoliko kHz. Razlog lezi u tome sto sto su nize frekvencije mozemo bolje odabrati opseg i to bolje slusamo, sto su vise frekvencije slabije cujemo pa su i veci opsezi. Svaki opseg dodje po jedan predstavnik naseg mozga. Znaci svih 20000 Hz koliko mi cujemo, 20000 frekvencija se uzorkuje na 25 vrednosti koje su najjace po amplitudi. Posto imamo dva uha, kako da znamo sa koje strane dolazi zvuk? Jednostavno, jedno uho ce dobiti pre drugog. To je interauralTimeDifference (ITD). Ako je ta razlika manja od 50 milisekundi, nas mozak ce zakljuciti sa koje strane dolazi zvuk. Ako je veca od 50 milisekundi, nas mozak ce registrovati kao da dolazi sa 2 zvucna izvora. Osim ITD, ima i razlika u amplitudi. Ono uho koje je blize izvoru ce glasnije cuti, jer zvuk udara u glavu pa se amplituda smanjuje.

Culo dodira je koncetrisano u covekovoj kozi. Najkasnije ga dobijamo, a ujedno ga i prvog gubimo. Imamo dva osecaja koja mogu da se registruju putem cula dodira. Prvi je osecaj pritiska, a drugi je osecaj vibracija. Moze se reci osecaj, a moze se reci i detekcija. Putem prstiju se najcesce ostvaruje detekcija vibracija. To se ostvaruje tako sto prstima vrsimo prevlacenje po povrsini i tom prilikom nam nabori na jagodicama vibriraju. Sto oni finije vibriraju, to je povrsina gladja, sto oni manje vibriraju (manja frekvencija), to je povrsina grublja. Detekcija pritiska se ostvaruje putem stiska. Prvi je hapticki, a drugi taktilni.

Na koji nacin covek vrsi obradu informacije, odnosno na koji nacin covek reaguje na dogadjaj. Prvo prima nadrazaja okoline putem cula. Potom, posto promenu pritiska slusamo, elektromagnetne signale gledamo itd, vrsi prevodjenje, kodiranje svih tih nadrazaja u bioelektricne signale koji se putem zivaca prenose do mozga. Kada dodje bioelektricni signal u glavu, vrsi se komparacija da se utvrdi o cemu je rec. Ako je putem komparacije nastala neka nasa reakcije, onda cemo formirati sta trebamo da uradimo i onda mozak, opet putem zivaca, naredjuje da se neki misici pokrenu, urade neku akciju, i time uticemo na okolinu.

Prethodno opisano predstavlja ugao automaticara. Medjutim, medicinari su ga napali i rekli da bez memorije i paznje, data komparacija ne bi bila moguca.

Sa stanosta HCI, odnosno nas tehnicara, proces obrade informacije coveka ide na sledeci nacin. Prvo covek prima razlicite nadrazaja okoline koji dolaze do razlicitih cula. Svako culo ima svoju memoriju koja se zove senzorska memorija (baferi). Desno oko ima svoj bafer, levo oko ima svoj bafer. Desno uho ima svoj bafer, levo uho ima svoj bafer itd. U bafere stalno pristizu informacije. Medjutim, mi ne mozemo reagovati i obradjivati sve informacije. Zato je jako bitna paznja. Paznja coveka tera da one informacije koje su u tom trenutku interesantne pokupimo iz bafera i obradimo. Ostale se odbacuju. U HCI je paznja bitna da bi te informacije koje se prikazuju u okviru naseg softvera i koje su vazne on sigurno vidi i obradi, tacnije da ih ne propusti. Nakon toga informacije koje su prosle "filter" paznje, dolaze do radne memorije. U datoj memoriji ih covek obradjuje. Pri toj obradi moze da obradjuje samo te informacije koje su dosle od cula, a moze i da trazi informacije iz trajne memorije. Takodje, nekada je potrebno izvrsiti ponavljanje obrade neke informacije u radnoj memoriji, radi boljeg razumevanja i pravog memorisanja u trajnu memoriju. Naravno, u radnoj memoriji moze da se vrsi i odbacivanje odredjenih informacija koje onda budu zaboravljene. Kada je u pitanju trajna memorija, ona informacije cuva trajno i to na dva nacina, odnosno dve strane. Prva je proceduralno i to su vestine (pevanje, plivanje, sviranje itd.). Druga strana jeste deklarativna koja moze biti semanticka i epizodna. Deklarativno semanticko pamcenje se bazira na pamcenju znacenja i znanja, a epizodno na dogadjajima, vremenima i mestima.

Senzorska memorija se moze najlakse opisati kao round-robin memorija, u koju pristizu informacije sve vreme i kada se ona napuni do kraja, jednostavno dolazi do gazenja onih prvih koji su pristigli. Informacije koje prodju "filter" paznje u okviru senzorske memorije odlaze dalje ka radnoj memoriji.

Radna memorija delimicno duze cuva informacije nego senzorska. Smatra se da je to negde oko 0.2 sekunde. Takodje, vazno je istaci i to da se u radnoj memoriji ne moraju obradjivati samo one informacije koje pristizu iz senzorske, vec mogu i one koje se iscitaju iz trajne memorije. Smatra se da je kolicina informacija koje mogu istovremeno da se obradjuju u okviru ove memorije otprilke 7 plus/minus 2.

Poslednji vid memorije jeste trajna memorija. Ona sluzi za trajno skladistenje informacija. Neki ljudi tvrde da sve informacije koje se uskladiste u ovu memoriju u njoj trajno i ostaju. Postoje dve vrste date memorije, to su epizodna trajna memorija i semanticka. Epizodna, kao i senzorska, predstavlja neki vid round-robin memorije samo duze. U nju se smestaju razliciti dogadjaji, mesta, iskustva itd. Semanticka trajna memorija sluzi za skladistenje cinjenica, sustina, fakta itd. Da bi nesto otislo u semantciku memoriju i tamo se sacuvalo potrebno je da se prave razlicite stvari, razlicite strukture. Prva takva stvar jeste semanticka mreza. To je najcesni nacin za memorisanje u sematicku memoriju. Bazira se na tome da se pamcenje bazira na shvatanju veza koje postoje po bilo kakvim elementima. Drugi nacin za sladistenje u semanticku mrezu jeste pomocu okvira i slotova. Zasniva se na tome da za svaki element koji je definisan jednim okvirom definisemo slotove koji ga opisuju. Ti slotovi jesu glavne osobine datog elementa, kao i ono sto ga odvaja od drugih okvira. Tu su jos i skripti, i IF THEN pravila.

Postoji 4 tipa paznje. Fokusirana paznja predstavlja pracenje jednog dogadjaja, jedne stvari iz grupe ostalih. Deljena paznja jeste ona koja omogucavanja pracenje vise dogadjaja. Dobrovoljna jeste ona koja se zasniva na volji, interesovanju. Poslednja je nehoticna gde najmanji dogadjaj moze da preuzme paznju.

Postoji nekoliko nacina na koje covek vrsi rezonovanje stvari. Prvi nacin rezonovanja jeste induktivno. Bazira se na generalizaciji. Primer jeste da znamo da slon ima surlu, pa kada nam neko spomene slona onda cemo znaci da svi slonovi imju surlu, iako nismo videli sve slonove. Deduktivno rezonovanje se bazira na premisama, odnosno zakljucivanju. Primer, "ako pada kisa, tlo nije suvo". Abduktivno se bazira na tome da kada nesto dozivima, onda smatramo da ce to uvek biti tako.

Takodje, postoje tri teorije koje kazu na koji nacin resavamo probleme.

Gestalt teorija kaze da covek resava probleme na dva nacina, reproduktivno i produktivno. Reproduktivno se bazira na resavanju problema putem necega sto smo culi, sto smo naucili, sto smo videli, neki kazu genetski preneli znanje sa nasih predataka na nas pa znamo. Produkivno se bazira na tome da mi produkujemo razlicite stvari.

Problem space teorija je sledeca. Ona prica o tome da covek resava probleme tako sto ih rasturi na mnogo malih. Resavanje problema se onda zasniva na resavanju tih malih potproblema. Problem sa ovom teorijom jeste na koji nacin da se vrsi dekompozicija. Moze doci do prevelikog broja manjih problema cija povezanost bude previse kompleksna. Takav i interfejs treba da bude. Treba se omoguciti da on daje coveku da resavaja jedan po jedan zadatak da dodje do resenja.

Teorija analogije je poslednja. Kaze da neki covek u nekom svetu je za resavanje nekog problema koristio neke operacije i on je sposoban da prebaci te operacije u drugi svet radi resavanja nekog problema. Sto je nova oblast bliza nekog oblasti koju poznajemo, analogija se lakse uhvati i obrnuto.

Ima dve vrste eksperata. Prvi su eksperti na senzorsko-motornom nivou. Bazira se na tome da covek na misic nauci nesto da radi. Eksperti na kognitivnom nivou se baziraju na poznavanju necega na

semanticko/epizodnom, vec i dugoj praksi u kojoj su znanja iz trajne memorije primenjivana. Za svaki posao, da bi postalo ekspert na kognitivnom nivou, potrebno je uraditi proceduralizaciju pa generalizaciju.

Postoji nekoliko faza sticanja vestina. Kognitivna faza predstavlja fazu sticanja vestina u kojoj smo apsolutni pocetnici. Koristimo svu dostupnu semantiku i jedini cilj nam jeste da zavrsimo to nesto sto smo zapoceli. Kako sticemo iskustva, tako pravimo nase procedure, vrsimo proceduralizaciju. Tako se prelazi u asocijativnu fazu. Gde vise koristimo nase procedure, a manje ono sto smo na pocetku naucili. Sada ima znanja i prakse koje ga teruju da postane efikasniji u primeni tog znanja. Guramo vestinu gore da bi postali efikasniji. Tako se prelazi u autnomnu fazu. Kada dodjemo u ovu fazu onda imamo pravo da kazemo da smo ekspert. U autonomnoj fazi smo sve procedure koje znamo usavrsili da budemo efikasni.

Postoje dve vrste eksperata. Prvi su eksperti na senzorsko-motornom nivou. To su eksperti koji su nesto naucili na "misic". Na osnovu prevelikog broja uradjenih stvari, recimo kucanja poruka, mogu vrlo brzo da odrade to nesto. Druga vrsta eksperata jeste ona na kognitivnom nivou. Eksperti na ovom nivou su oni koji pored dosta semantickog i epizodnog znanja imaju i dosta prakse, iskustva sa time sto rade. Za to trebaju godine iskustva. U svakom procesu postajanja eksperta na kognitivnom nivou ide prvo proces proceduralizacije pa generalizacije. Ta proceduralizacija i generalizacija predstavljaju proces dobijanja veceg skill-a. Sticanje vestine se bazira na postojanju i prelasku 3 faze. Prva faza jeste kognitivna faza. U datoj fazi smo kada smo apsolutni pocetnici. Onda koristimo svu semantiku koju smo pribavili. Kada smo u ovaj fazi cilj nam jeste samo da sto pre to nesto zavrsimo. Nit nam je cilj da bude efikasni, niti koliko cemo gresiti, nego samo da zavrsimo. Kako se skuplja iskustvo, povecava skill u toj nekoj oblasti, tako dolazi do pravljenja nasih sopstenih procedura, kao nekih oblika malih odstupanja od pravila. Sada koriscenje tih procedura nas vodi ka resenju. Proceduralizacijom se prelazi u drugu fazu. Druga faza je oznacena kao asocijativna. Zove se asocijativna jer mi pravimo razlicite veze, asocijacije izmedju razlicitih malih zadataka i pravimo nase procedure. Sada kada vec ima svoje procedure, covek postaje zeljan da na osnovu znanja koje poseduje i iskustva postane efikasniji u primeni tog znanja. Na taj nacin se obavlja generalizacija i prelazak u poslednju fazu, autonomna faza. U autonomnoj fazi smo sve procedure prosirili, usavrsili, namontirali da budemo efikasni. Prilagodjavamo se maksimalno situaciji uz ostvarivanje maksimalne efikasnosti. Kao izvor informacija su najbolji ljudi u kognitivnoj fazi. Za testiranje su najbolji ljudi u autonomnoj fazi, jer su upoznati sa oblasti i mogu nam pomoci vrlo lako. Najgora i najopasnije greske se prave u poslednjoj fazi. Razlog lezi u tome sto smo mi generalizaciju napravili uvek na istoj okolini. Dovoljno je da se u toj okolini desi nesto neubicajeno i nastane katastrofa. Taj tip greske se naziva slip. Drugi tip gresaka jeste mental models. Mentalni model jeste skup pretpostavki kako nesto radi. Mentalna slika predstavlja jednu stanje mentalnog modela, odnosno zamrzuti mentalni model u jednom trenutku. Mentalna slika je veoma vazno, jer preko interfejsa mi trebamo da damo u covekoj glavi mentalni model koji mu govori na koji nacin nesto radi, cemu nesto sluzi. Ima 3 vrste mentalnih modela. Prvi je model sistema. To je pravi model. Ako pricamo o televizoru, to je njegova elektricna sema. Mentalni model je sledeci. To je shvatanje korisnika na koji nacin nesto funkcionise. Projektantski model je model koji je projektant napravio da se taj uredjaj prikaze coveku. Dobar interfejs jeste ako projektantski model se sudari sa korisnikovim modelom. Medjutim, ima jako korisnika sa jako puno modela i to nije bas mnogo lako napraviti. Sva modela trebaju biti jednaka kada pravimo softver za nase kolege.

TASTATURA:

Tastature su namenjene da covek unosi tekst putem njih. Zasniva se na principu: jedan simbol, jedan pritisak, jedan taster. Proizvodjaci su se jako trudili da osmisle koliko da bude velik taster. Ne sme biti prevelik, jer ce biti prevelika tastatura, pa ce se covek zamarati. Takodje, ne sme biti ni previse mali,

jer onda bi doslo do cestih gresaka proizvedenih istovremenim klikom vise tastera istovremeno. Razmisljaji su i o tome koliko covek treba jako da pritisne taster. Opet ne sme biti premalo, jer onda covek ni nema utisak da je kliknuo. S druge strane, ne sme da bude ni prejako, jer se time covek opterecuje vise na kliktanje, nego na tekst koji treba da unese. Vodilo se racuna i o tome koliko treba da bude hod tastera da bi korisnik imao osecaj da ga je pritisnuo. Doslo se do zakljucka da je negde 3-4 mm dobar hod, odnosno da je pri tom hod dobar osecaj pri kucanju. Kada dodje do dna da korisnik ima osecaj da je kliknuo. Ne treba da se cuje, nego da se oseti na jagodicama klik. Naravno, bitan je i raspored tastera na tastaturi. Pre svega, postoji QWERTY raspored. Raspored se zasniva na frekvenciji pojavljivanja slova jednih pored drugih na engleskom govornom podrucju. Takav raspored je dodatno izmenjen kako se mehanika te masine ne bi zaglavljivala cesto. Postoji i DVORAK raspored. Dati raspored je izbacio opterecenost mehanikom. Takodje, imamo i ABCDE raspored. Imamo nekoliko vrsta tastatura. Prve tastature su bile ravne i terale su da ruke, laktovi budu uz telo. Izuzetno je zamarajuce, pa se i tezilo ka odvajanju laktova od tela. To je bilo dovoljno da se napravi ERGONOMSKA tastatura. Ona se dobija tako sto se presece stara tastatura te se dobiju dve polovine. Jedna za jednu ruku, druga za drugu ruku. Samim tim nas nateralo da mozemo ruke drzati razdvojenim sto je daleko rasterecujuce. Takodje su je i podigli malo gore kako bismo imali mesta da stavimo ruke ispred nje. Problem date tastatura lezi u tome sto je neophodno poznavanje slopog kucanja. Sve prethodno recenu su fizicke tastature. Za razliku od njih, danas imamo i virtuelne ("on screen") tastature. Prvo pocele u telefonima, danas se prebacuju i na racunare. Prednost je u tome sto se vise ne mora pritiskati, nego da se moze i pomerati prst za unos simbola. Takodje, moguce je pomeranje slova. Naravno, nema ni dodatnih uredjaja (tastature). Losa stvar jeste slab "feedback".

Fizicka prednosti:

- bolji feedback (hod tastature, velicina tastera, oblik)
- manje zamorna za koriscenje

Fizicke mane:

- fizicki uredjaj
- ne mozemo da menjamo raspored
- mogucnost kvara

"On screen" tastature prednosti:

- mozemo da menjamo raspored
- manji prostor (ne zauzme fiziki prostor, ali zauzme ekran)
- moguc unosa prevlacenjem

"On screen" tastature mane:

- losiji feedback (vibracija, zvuk, slicica)
- zauzima ekran
- teska je za koriscenje svaki dan, ceo dan (brzo zamaraju, nemaju oslonac za ruke)

POINTERSKI UREDJAJI:

Pointerski uredjaji su oni pomocu kojih se vrsi proces ukazivanja, selekcije, direktnu manipulacije. Usko su vezani za WIMP. Postoje dve vrste pointerskih uredjaja. Prva vrsta su pointerski uredjaji direktne kontrole. To znaci da se oni direktno primenljuje na prikazne uredjaje. U takve uredjaje spadaju svetlosno pero, prst, digitalna olovka. Direktno ukazujem na to nesto na ekranu. Druga vrsta pointerskih uredjaja su indirekne kontrole. U takve uredjaje spada mis, trackball, joystick, touchpad itd. Pointerski uredjaji

indirektne kontrole su oni koji se ne primenjuju direktno na prikaznom uredjaju, nego po nekoj drugoj povrsini tako da njegov posrednik dodje tamo gde treba.

Direktna kontrola prednosti:

- prirodnije je za koriscenje, jednostavno na nesto pokazem, intuitivniji su
- preciznije je u smislu slobodnog pokreta (laksi za crtanje)
- manje zauzimaju prostora za upotrebu

Direktna kontrola mane:

- losiji su kada je widget / target / cilj manji
- vece je zaklinjanje povrsine (prikaznog uredjaja)
- vise zamorno

Indirektna kontrola prednosti:

- bolji su kada je widget / tager / cilj manji
- manje je zakljinjanje povrsine, ekrana (zakljinjemo onoliko koliko je kursor)
- manje zamorno

Indirektna kontrola mane:

- nije toliko prirodno za koriscenje, manje intuitivni
- manje precizno
- vise zauzima prostora za upotrenu

Kada su u pitanju ekrani osetljivi na dodir, imamo nekoliko vrsta. Prvi su REZISTIVNI/otporni. Na obicne ekrane nalepe se sendvic folije. Njih cine providna folija, koja je podeljena na male kockice/targete, ispod svake kockice nalazi se jedno malo jastuce od materijala koje je elasticno. Ako nije sabijeno ima jednu otpornost, ako se malo sabije ima drugu otpornost. Nakon toga se nalazi drugi red folije. Selektovanje se bazira tako da moram da pritisnem taj ekran, dolazi do promene otpornosti i zna se sta je kliknuto. Nije savrseno transparentan, gusi svetlost. Samim tim je potrebno pojacati osvetljenje ekrana. Pojacanjem osvetljenja ekrana, brze se trosi baterija. Prvi telefoni su imali takve. Na bilo kom uredjaju u intendustriji se koristi. Otporni su na spoljasnost. Za razliku od njih, u telefonima se koriste KAPACITIVNI ekrani osetljivi na dodir. Bazira se na kondenzatorima. Koliko je rezolucije, toliko konzendezatora. Pod rezolucijom se ne misli rezolucija ekrana, nego rezolucija za dodir. Nema pritiska, nego je dovoljno samo da ga pipnem. Radi po principu da je kondenzator napunjen elektricitetom i na nas dodir dolazi do praznjenja datog kapaciteta kondenzatora. Kapacitivni ekran je potpuno provodon, ne gasi svetlnost. Minus je taj sto je osetljiv.

Postoji tri vrste miseva. Mehanicki, opticki i laserski. Mehanicki se bazira na gumenoj lopti koja se krece po povrsini. Problem je usput izigrava i usisivac. Zbog toga se preslo na laserske. Problem kod laserskog je u tome sto nece dobro raditi ukoliko podloga nema na sebi tu boju koja je boja lasera. Recimo crvena boja lasera i crvena boja podloge dovesce do toga da crvena boja upije taj laser. Zbog toga je napravljen opticki. Zasniva se na kameri koja snima otsjaje povrsine, to belezi i na taj nacin prepoznaje pomeraje.

Trackball se bazira na loptici koja je ugradjena tastaturi najcesce. Spada u najpreciznije pointerse uredjaje. Pored toga sto je jednostavan, dobar je jer i zauzima fiksno jednu poziciji. Omogucava smanjenje homing-a.

Joystici su preuzeti iz vojske. Nije bas za svaku primenu. Za kucanje u wordu i navigaciju nije bas

najbolji.

Tracpoint se bazira na gumi. Pomeranjem gume se detektuje gde zelimo da poentiramo. Ugradjuje se u tastaturu i prednost je ta sto je izuzetno mali. Tactile trackpoint omogucava slepima da bukvalno napipava sadrzaj ekrana pomocu iglica. Jako osetljivo, brzo se kvari te nije ni zazivelo.

Toucpad radi kapacitivno.

FICOV ZAKON:

- vreme poentiranja je vece sto je veci put izmedju pocetne i ciljne tacke koju gadjam
- sto je cilj dalje, to je potrebno da cilj bude veci i obrnuto.

MONITORI:

Sluze kao prezentacioni, prikazni uredjaji, sluze za prikaz.

CRT monitori se zasnivaju na fosforu. Fosfor je taj koji daje boju pikselu. Najbolji je po lepoti boja. Jako je brz, u smislu da moze da podrzi brze promene. Ugao gledanja mu je veliki. Lose kod CRT jeste taj sto je "debeo", sto je veci ekran to je i on deblji. Losa strana mu je i to sto je najveci potrosac struje od svih monitora.

CRT opste:

- zasniva se na fosforu

CRT prednosti:

- izuzetno lepe boje pruze putem koriscenja fosfora
- ugao gledanja mu je izuzetno velik
- brz je, moze da isprati brze promene slika

CRT mane:

- sto je veci ekran, to je i zauzima vise prostora sto se tice debljine
- izuzetno je velik potrosac struje

Plazma monitori se zasnivaju na fosforu. Dobar ugao gledanja. Bolji od LCD, losiji od CRT. Bazira se na tome da svaki piksel ima tri male posudice, gde je dno namazano fosforom odredjene prljavstine. Izuzetno je mali potrosac. Omogucava ogromne velicine ekrane. OLED samo moze imati vece ekrane. Minus se ogleda u tome sto je duzina trajanja izuzetno mala. Razlog lezi u tome sto se tanki namazi fosfora brzo potrose. Takodje, opasnost jeste i curenje plemenitog gasa koji sprovodi struju. Kada ga nestane ode piksel. Prednosti su to daje dobre boje, ugao gledanja dobar, velike dimenzije, a tanak.

PLAZMA opste:

- zasniva se na tome da svaki piksel ima tri male posudice cije je dno namazano fosforom odredjene prljavstine
- koristi se i plemeniti gas kao sprovodnik struje

PLAZMA prednosti:

- izuzetno lepe boje, bolje nego LCD, losije nego CRT
- ugao gledanja mu je velik

- izuzetno mali potrosac
- omogucava ogromne ekrane, samo OLED pruza vece
- tanak

PLAZMA mane:

- krataka vek trajanja, jer se namazi fosfora brzo potrose
- opasnost jeste i curenje plemenitog gasa, te se onda ne moze struja provoditi i dolazi do otkazivanja piksela

LCD se ne baziraju na fosforu, vec na propustanju svetlosti, odnosno na zaokretanju svetlosti. Bazira se na tecnom kristalu koji propustaju tudju svetlost putem zaokretanja. Kaze se tecni jer njegovi kristali nisu cvrsto vezani jedni za drugo, nego su izuzetno labavi. On samo propusta ili ne propusta tudju svetlost. Postoji standardni LCD i LED LCD. Nije brz, ne moze da podrzi brze promene slika. Sve ce biti zamazano. Los ugao gledanja. Izuzetno jeftin. Mali potrosac. Tanji od CRT.

LCD opste:

- zasniva se na tecnom kristalu koji propusta tudju svetlost putem zaokretanja datih kristala

LCD prednosti:

- izuzetno jeftin
- tanak, tanji od CRT
- mali potrosac

LCD mane:

- nije brz
- lost ugao gledanja
- nema svoju svetlost

OLED, odnosno organski LED, se bazira na lampicama, gde jedna lampa predstavlja jedan piksel. OLED, za razliku od LCD, ima svoju svetlost. Malo su losije boje od fosfora, ali bolje u odnosu na LCD i eink. Dobar je ugao gledanja, ne kao kod CRT ali bolji od LCD. Izuzetno su tanki, a mogu da pokriju citav zid. Cena je najveca od svih. Brz, mala potrosnja, savija se, sve je top.

OLED opste:

- bazira se na lampicama, gde jedna lampica predstavlja jedan piksel

OLED prednosti:

- losije boje od fosfora, bolje od LCD i eink
- dobar ugao gledanja, ne kao kod fosora ali bolje od LCD
- brz
- mala potrosnja
- ogromni ekrani

OLED mane:

- izuzetno skup

Eink monitori se baziraju na elektronskom mastilu. Ni on nema svoju svetlost, vec samo reflektuje spoljnu svetlost. Najmanji potrosac. Baterija moze trajati i do mesec dana. Mana se ogleda u tome sto moze da prikazuje samo nijanse boje koje je mastilo (monohrono). Treba tudje svetlo.

EINK opste:
- zasniva se na elektronskom mastilu
- nema svoju svetlost, vec samo reflektuje spoljasnju

EINK prednosti:
- najmanji potrosac

EINK mane:
- treba tudja svetlost
- monohroni prikaz boja

Rad kamere se bazira na tome da svetlost pada na video cip. Video cip moze biti napravljen na dva nacina, CCD i CMOS. CCD se bazira na dva para tranzistora. Jako je dobar pretvarac svetlosti u elektricni signal. Nevolja se ogleda u tome sto je po povrsini velik. Tako da na jednom cipu nije moguce napraviti veliku rezoluciju. CMOS se bazira na jednom tranzistoru. Po povrsini je daleko manji od CCD. To znaci da na jednom video cipu da ih se nagura koliko hoces, sto dalje implicira veliku rezoluciju. Mana je u tome sto je losiji, nekvalitetniji.

CCD opste:
- dva para tranzistora ga cine

CCD prednosti:
- dobar pretvarac svetlosti u elektricni signal

CCD mane:
- po povrsini velik, tako da se ne moze napraviti po jednom cipu velika rezolucija

CMOS opste:
- jedan tranzistor

CMOS prednosti:
- nekoliko puta manji od CCD, sto znaci da na jednom video cipu moze jako puno da ih stane, rezolucija koliko hoces

CMOS nedostaci:
- losiji od CCD, jer kada se snima nesto sa njim slabije se prilagodjava na nove okolnosti / promenu okruzenja

HDR se bazira na tome da napravi vise slika u kratkom vremenu. Gde se svaka slika napravi sa manjim otvorom blende. Nakon toga putem algoritma uzme najbolje detalje iz svake slike. Mora biti miran uredjaj i uvek se gube detalji (algoritam nije savrsen).

Jedni e-copy uredjaji su monitori, zato sto prikazuju elektronsku verziju dokumenta, dok su drugi skeneri. Oni hardverski zapis pretvaraju u elektronski. Oni to postizu skeniranjem. Postoji dve vrste

skenera, refleksivni i transparentni. Problem refleksionih je taj sto pomocu njih ne mogu da se skeniraju rengenski snimci. Razlog lezi u tome sto ce oni upiti svu svetlost. Zbog njih su nastali ovi drugi, transparentni.

Dve vrste skenera: refleksivni i transparetni.

Postoji dve vrste bim projektora, LCD i DLP sa MEMS cipovima. LCD su veci, sporiji, jeftini i nemaju tako veliku moc da osvetle. Razlog loseg projektovanja svetla se bazira na tome da je potrebno da svetlo prodje kroz LCD panel. Potreban je veci mrak da bi slika bila lepo prikazana. DLP se bazira na MEMS cipovima. Brzi je od LCD, bolji prikay, ali i dosta skuplji.

Hard-copy prikazu su oni koji, za razliku od e-copy, vrste prikaz na papiru. Stampaci su hard-copy jer oni ostavljaju kopiju na papiru. Prvi su impact line stampaci. Radili su po principu pisace masine. Izuzetno brz, to je prednost. Mane se ogledaju u tome sto moze stampati samo onom bojom koja je mastiljava traka i to sto moze stampati samo alfanumerike, nema grafike. Takodje, pise samo jedno pismo. Sledeci su dot matrix. Baziraju se na ostavljanju tacaka na papiru. Mogao je i da crta. Manje bucan, ali znacajno sporiji. Naredni su termalni. Tihi su jer nema lupanja kao kod prethodna dva. Omogucava i prikazivanje boja. Bazira se otpornicima koji, kada se zagreju, peku papir. Naravno, koristi se specijalni papir. Isti kao dot matrix, samo moze u boji. Poslednji su inkjet stampaci. Bazira se na pljuckanju boja na papir. Najbolji je prikaz grafike. Postoje dve vrste inkjet stampaca, to su termalni i piezoelektricni.

IMPACT LINE prednosti:
- izuzetno brz

IMPACT LINE mane:
- samo onom bojom koja je boja mastila
- stampa samo alfanumerike, nema grafike
- izuzetno bucan

DOT MATRIX prednosti:
- moze grafika
- manje bucan

DOT MATRIX mane:
- sporiji