Dankwoord

Graag zou ik mijn promotor, Prof. Dr. Olga De Troyer bedanken voor de verbeteringen, opmerkingen en hulp die zij mij gegeven heeft. Mede door haar goede raad heeft ze me op het goede spoor gezet.

Koen De Hondt, Johan De Geyter en Patrick Steyaert van MediaGeniX wil ik bedanken voor hun suggesties en advies over het onderwerp van dit eindwerk.

De werknemers van het computerbedrijf 'Namahn' dank ik voor het gebruik van hun bibliotheek waar ik veel belangrijke informatie heb vernomen.
Dhr. Joannes Vandermeulen, hoofd van business development, wil ik in het bijzonder bedanken voor zijn deskundig advies.

Hierbij zou ik ook mijn dank willen betuigen aan mijn medestudenten, Gregory Vandenbroucke en David Van Damme, voor hun steun, suggesties en uitleg bij bepaalde kwesties.

Tenslotte nog een dankwoord aan mijn ouders, broer en grootouders voor hun steun en zeker niet te vergeten aan Sabrina, mijn vriendin, voor het nalezen en verbeteren van de tekst.

Inhoudstabel

1 Introductie

De wereld van netwerken heeft drastische veranderingen ondergaan de laatste 5 jaar. LAN's zijn geëvolueerd van 10Mbps naar 1Gbps, dus 100 keer in snelheid gestegen, en met de opkomst van breedband WAN's die betaalbaar zijn voor iedereen, zoals kabel en ADSL, is de mogelijkheid ontstaan om ook voor het Internet geavanceerde gedistribueerde systemen te ontwikkelen. Hierbij denken we o.a. aan gedistribueerde virtuele omgevingen, waarvan er reeds enkele in gebruik zijn. Technologieën zoals VRML ("*Virtual Reality Modelling Language"*) en VRTP (*virtual reality transfer protocol*), naar analogie met HTTP en HTML (die we sectie 3.2 zullen bespreken) wijzen er op dat het Internet bruikbaar is voor volwaardige applicaties, en niet enkel voor het uitwisselen van informatie. Het Web wordt dynamischer, interactiever en kneedbaarder.

Cooperative computing is een belangrijk paradigma om verschillende partijen samen te laten werken om een vooropgesteld en niet triviaal doel te bereiken. Het omvat belangrijke aspecten zoals *computer supported cooperative work* (CSCW), *workflow*, *computer assisted design*, databanken en *concurrent programming*. Het Internet heeft een revolutionaire verandering teweeggebracht, namelijk het voorzien van een geünificeerde transportinfrastructuur voor netwerken. Gescheiden computers kunnen nu onmiddellijk participeren in gedistribueerd *computing*. Zo'n raamwerk maakt het gebruikers mogelijk om naadloos met anderen samen te werken en te communiceren over geografisch gescheiden plaatsen. Aangezien technologieën evolueren en geavanceerder worden is er een toenemende nood aan onderzoek en ontwikkeling van toepassingen en applicaties om te genieten van de voordelen van Internet en het Web om zo gebruikers en programmeurs een interactieve en robuuste coöperatieve computing omgeving te bieden.

Er zijn wel enkele nadelen aan werken met het Web als communicatiemiddel. De transfersnelheid ligt bijvoorbeeld nog relatief laag, data kan verloren gaan en de functionaliteit van de technologieën is beperkter, vaak wegens veiligheidsredenen. De meeste applicaties die werken op het Internet zijn bijvoorbeeld opgebouwd volgens het *client/server* model. In dit werk gaan we onderzoeken welke nadelen er zoal zijn en hoe deze invloed hebben op het ontwikkelen van zo'n collaboratieve applicaties, welke eventuele oplossingen er mogelijk zijn en wat er haalbaar is op het WWW.

Er zijn vier grote delen in deze thesis, die als bedoeling heeft aan te tonen in hoeverre we collaboratieve applicaties we kunnen bouwen in de *WWW Shell*. De WWW Shell is de verzameling van technologieën en applicaties die betrekking hebben tot het WWW.
In hoofdstuk 7 spitsen we ons toe op het collaboratieve deel van heel dit onderzoek. We bekijken eerst welke raamwerken er zijn om groupware op te splitsen, om zo de verschillende belangrijke verschilpunten tussen de systemen onderling te kunnen inschatten. Nadien bekijken we de (belangrijkste) verschillende types van collaboratieve systemen die er bestaan, om dan de implementatieaspecten te bekijken.
Vervolgens, in het derde hoofdstuk, bekijken we de geschiedenis van het Internet en spitsen we ons toe op de voornaamste webtechnologieën en hun specifieke problemen en eigenschappen. Ook de. We eindigen dit hoofdstuk met de aspecten waarmee we moeten rekening houden bij de drie grote delen van het Web, de clients, de server en het netwerk.
In het vierde hoofdstuk voegen we als het ware alles samen en onderzoeken we welke groupware systemen haalbaar zijn op het WWW [WWW]. We onderscheiden degene die al bestaan, die dat nog niet gebouwd zijn maar haalbaar zijn en die dat (nog) niet haalbaar zijn op het WWW.
En in het vijfde en laatste tenslotte voegen we alle bevindingen samen en ontwerpen we een totnogtoe niet bestaand collaboratief systeem, dat haalbaar zou moeten zijn over het *World Wide Web*.

2 Types van collaboratieve systemen

2.1 Introductie

Computer Supported Cooperative Work (CSCW) gaat over groepen gebruikers: op welke wijze systemen ontwerpen die hun werk als team ondersteunen en hoe het effect van technologie op hun werkpatroon te begrijpen. Het gaat dus verder op de bestaande kennis over een uitgebreide waaier van disciplines, onder andere over de sociale aspecten van het computeren.

Eén groot deel van het werk in CSCW bestaat in het voorzien van applicaties die groepen ondersteunen bij het samenwerken, de zogenaamde *groupware*. Groupware kan verschillende soorten van activiteiten ondersteunen:

- directe onpersoonlijke communicatie
- het genereren van ideeën en maken van beslissingen
- Het samenwerken aan (computer)objecten

Groupware applicaties worden vaak geclassificeerd op basis van de volgende criteria en:

- plaats en tijd.
- type van de gedeelde informatie.
- aspecten van de samenwerking die ondersteund worden.

Het ontwikkelen van groupware is moeilijker dan het ontwikkelen van applicaties voor één gebruiker, namelijk omwille van:

- netwerkvertragingen;
- veel mogelijke interacties;
- grafische *toolkits* om software interfaces te creëren veronderstellen één enkele gebruiker.

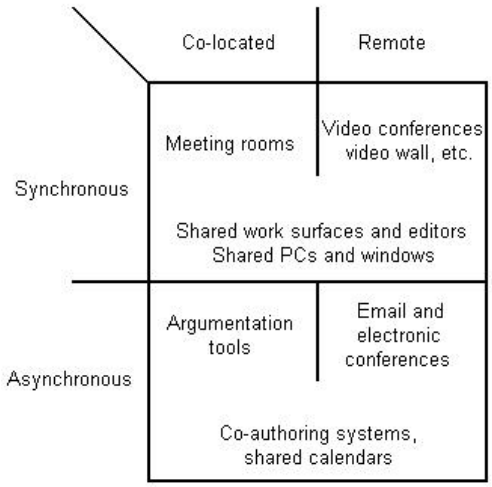
In dit hoofdstuk zal ik verschillende soorten groupware bespreken. Ook de verschillende raamwerken om groupware te klasseren en te plaatsen binnen de wijdere omgeving van coöperatief werken worden besproken. Vervolgens worden een aantal architectuur- en ontwerpkwesties behandeld. De meeste tekeningen en de grote lijnen in dit hoofdstuk komen vanuit het boek ‘Human Compter Interaction’ ([Dix et Al.98]).

2.2 Raamwerken voor groupware

In deze sectie zullen we verschillende raamwerken beschrijven die bijdragen tot het begrijpen van de rol van groupware. Deze zullen dienen om de groupware te classificeren, nieuwe applicatiedomeinen te ontdekken en bovendien kunnen ze het design van nieuwe systemen helpen structureren.

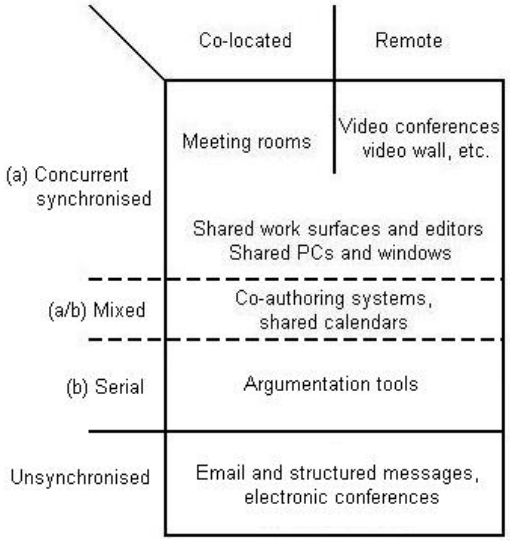
2.2.1 "Time/Space" matrix classificatie

Groupware kan op verschillende manieren geclassificeerd worden. Eén manier hiervan gebeurt volgens de plaats en het tijdstip dat de gebruikers hun taak volbrengen. Dit kan men voorstellen in een tijd/ruimte matrix, die een gevestigde waarde is geworden in de CSCW gemeenschap. Het is een 2x2 matrix, met op één as de tijd, namelijk synchroon en asynchroon, en op de andere de ruimte, *remote* en *co-located*. Het ruimteaspect is éénvoudig. Men beschouwt dat mensen op dezelfde locatie zijn als ze met elkaar *face-to-face* kunnen communiceren, anders zijn ze remote. Synchroon en asynchroon betekenen niet eenvoudigweg dat de gebruikers tegelijkertijd of niet werken. E-mail is immers een asynchrone applicatie en toch kan men er gelijktijdig naar elkaar schrijven. Het verschil ligt hem in het feit dat er bij synchrone applicaties op dezelfde data gewerkt wordt. Men moet dus elkaars veranderingen zo snel mogelijk kunnen zien. Een andere manier om het verschil te bekijken is dat asynchrone systemen zonder permanente computer connectie kunnen werken, andere niet.



Figuur 2.2-1: Tijd/ruimte matrix ([Dix et Al. 98])

We kunnen bovendien synchrone systemen nog in verschillende soorten onderverdelen. Er zijn de concurrente systemen, zoals videoconferenties en elektronische vergaderruimtes, waar mensen tegelijkertijd op dezelfde gegevens werken, en de seriële systemen, waar *locking* ervoor zorgt dat gebruikers dit niet kunnen. *Argumentation tools* vallen in deze categorie. Wanneer deze blokkeringmechanismen fijn genoeg werken, kunnen ze zowel serieel als concurrente toegang bieden, zoals bij kalenders en *co-authoring* systemen.



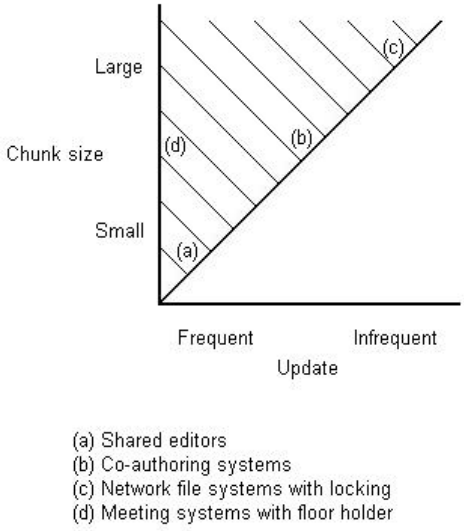
Figuur 2.2-2: Geraffineerde tijd/ruimte matrix ([Dix et Al. 98])

2.2.2 Classificatie gebaseerd op de gedeelde informatie

Een andere manier om groupware te classificeren is via de informatie die ze delen. Sommige systemen dienen voornamelijk om te communiceren, andere om met meerdere gedeelde gegevens te bewerken.
Hierbij kunnen we kijken naar de granulariteit waarmee informatie gedeeld kan worden, met andere woorden het niveau waarop informatie gedeeld wordt en de soort van objecten die gedeeld kunnen worden.

Granulariteit

De granulariteit is in dit type systemen zowel de frequentie van het vernieuwen als de grootte van stukjes objecten. Bijvoorbeeld, hoe vaak moet men in een tekstverwerker vernieuwen en gebeurt dit per woord, per zin of per alinea? De extremen zijn per letter en per bestand en de meeste systemen zitten hiertussen. De frequentie van vernieuwen heeft hiermee dan ook te maken. Als men per woord wilt veranderingen doorpropageren, moet de frequentie dus ook hoger liggen. Er is dus een wisselwerking tussen de grootte en de frequentie (zie figuur 2.2-3)

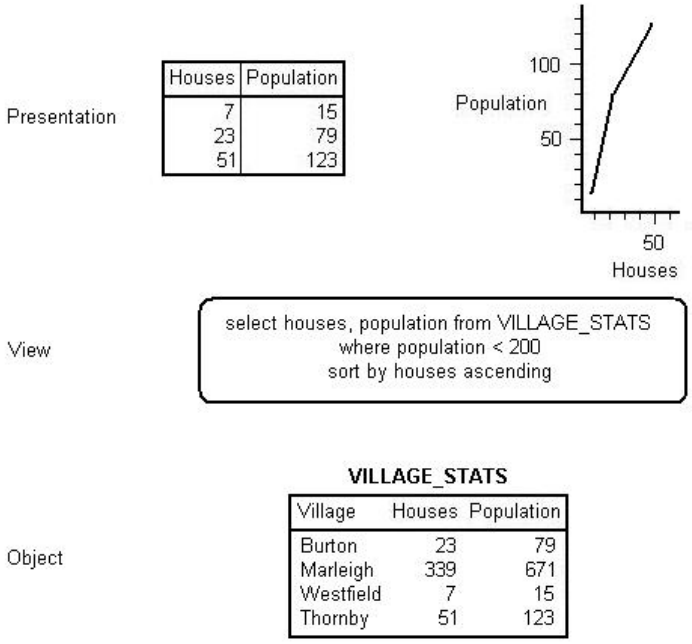


Figuur 2.2-3: Granulariteit van het delen ([Dix et Al. 98])

Niveaus van delen

Hier wordt eerder de nadruk gelegd in hoeverre systemen onderling verschillen in het opzicht van wat er gedeeld wordt. Eén extreem is het "What You See Is What I See" of WYSIWIS voorstelling, waarbij alle gebruikers dezelfde kijk op de informatie hebben. Dit kan een nadeel zijn in tekstverwerkers, waar gebruikers verschillende stukken tegelijkertijd willen aanpassen. Er is ook een interessante middenweg, waar verschillende gebruikers dezelfde informatie te zien krijgen maar niet dezelfde representatie van die informatie. Bijvoorbeeld één persoon wenst een grafiek te zien, de andere een taartdiagram. Bovendien kunnen we ook naar de niveaus van gedeeld gebruik kijken ten opzichte van de invoer: er is de mogelijkheid van een enkel insertiepunt (gedeeld keyboard) of van meerdere

insertiepunten (één per gebruiker). Bij deze laatste mogelijkheid is er ook nog het aspect van zichtbaarheid: zijn de andere gebruikers zichtbaar, zijn er dus meerdere insertiepunten of cursors zichtbaar en indien niet, is er dan een gedeelde cursor (*group pointer*)? Hier ook zijn er connecties tussen de systemen (bijvoorbeeld: het is niet logisch om verschillende gezichtspunten te hebben en slechts één insertiepunt).



Figuur 2.2-4: Niveaus van gedeeld gebruik ([Dix et Al. 98])

Soorten van objecten

Een ander logisch onderscheid die we kunnen maken tussen de verschillende systemen is volgens de objecten of data waarop we werken. Problemen treden vooral op bij de asynchrone systemen, waar er bij het synchroniseren van de data twee aanpassingen mogelijk zijn zonder dat we weten welke het eerst voorkwam.
Keuzes in het design kunnen gemaakt worden om zulke problemen te vermijden. Bij een klassieke tekstverwerker moet het systeem ervoor zorgen dat één gebruiker niets weghaalt waaraan een andere aan het werken is, maar bij een lineair tekstsysteem, waar de nieuwe alinea's eenvoudigweg aan het einde van de bestaande tekst worden geplakt, moet het systeem daarmee geen rekening houden. Hier is de tekst wel opeenvolgend, wat hem het meest geschikt maakt voor synchrone groupware, het probleem doet zich hier echter

voor wanneer twee personen tegelijk iets toevoegen. Elk ziet zijn eigen toegevoegde tekst als laatste staan. De twee personen krijgen niet meer dezelfde gegevens te zien. Dit kan gebeuren bij tekstgebaseerde communicatie, o.a. 'MSN Messenger' [Messenger] laat dit toe maar dit mag niet het geval zijn bij argumentation tools waar de volgorde van toevoegen van belang is. Hier kunnen we werken met tijdstempels, om zo te bepalen welke aanpassing er eerst kwam.
Shared Hypertext zonder aanpassen of verwijderen werkt met knopen, hier heeft het dus geen belang in welke volgorde contributies werden toegevoegd, de links ertussen zorgen immers voor de structuur. Maar stel dat die tekst wel aangepast kan worden en dus een complex gestructureerd object wordt, dan volgen de problemen. Wanneer bijvoorbeeld iemand een stuk Hypertext verplaatst terwijl iemand anders in een knoop werkt. Meerdere structurele aanpassingen zijn bovendien moeilijker dan meerdere tekstuele.

2.2.3 Classificatie gebaseerd op de functie van de groupware

Een andere classificatie is volgens de functie die het systeem ondersteunt, bijvoorbeeld een vergadering ondersteunen of het ondersteunen van *group authoring* (het schrijven met meerdere auteurs). We kunnen hier drie grote soorten onderscheiden:

- *computer mediated communication:*
  ondersteunt directe communicatie tussen de participanten
- *meeting and decision support systems:*
  moeten de gebruikers een gezamenlijk begrip bezorgen en opslaan
- *shared applications and artefacts:*
  helpen de gebruikers om samen op de gedeelde voorwerpen te werken.

2.3 Groupware systemen

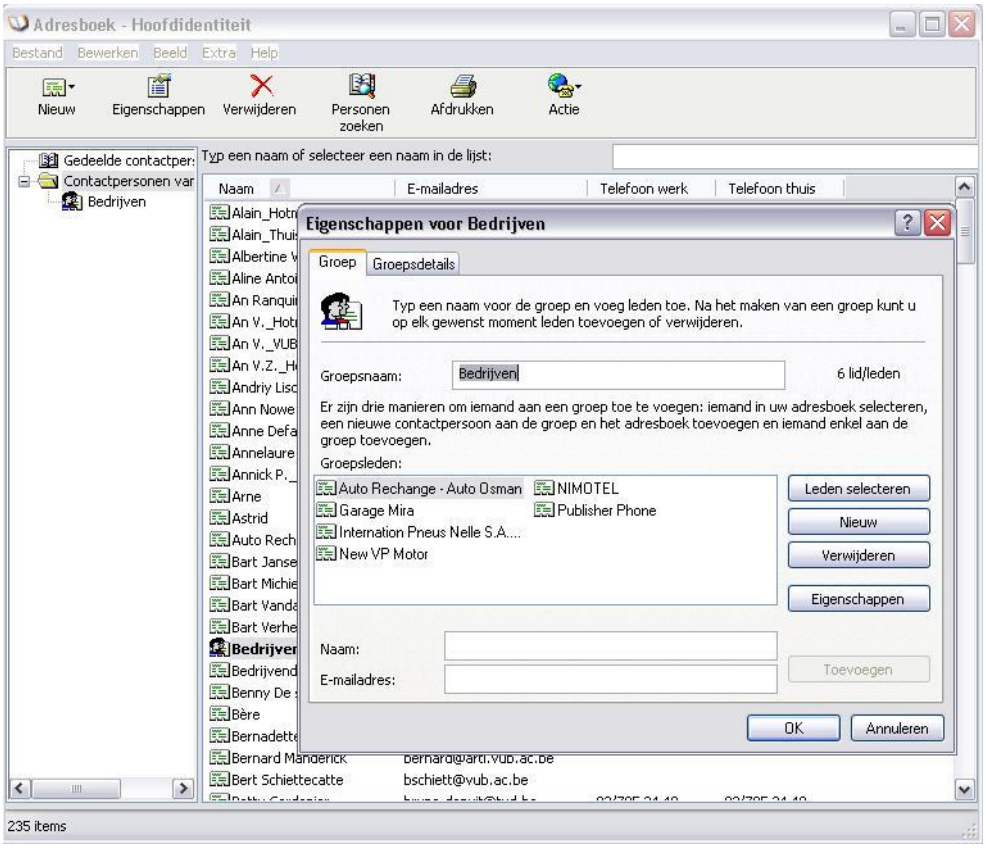
In de volgende paragrafen gaan we nadrukkelijker de verschillende soorten systemen bekijken, volgens de rangschikking uit 2.2.3.

2.3.1 Computer-mediated communication

Impliciet hebben we bij groupware twee of meer participanten die één of andere vorm van communicatie gebruiken. Systemen die enkel gecreëerd werden om deze directe communicatie te verzorgen noemen we *Computer-Mediated Communication* (CMC), een belangrijk onderdeel van CSCW. Voorbeelden van CMC zijn e-mail en *bulletin boards*, *structured message systems* en videogebaseerde systemen.

E-mail en bulletin boards

Iedereen heeft wel eens een vorm van e-mail gebruikt, en vele mensen kennen ook het bulletin board of forum.
Wanneer we een e-mail willen verzenden moeten we deze eerst opstellen en vervolgens instrueren om de e-mail te versturen. Een tijd later – dit kan gaan van enkele seconden tot verschillende dagen – krijgt de ontvanger jouw e-mail, wordt hij verwittigd en opent hij hem. Het probleem met e-mail is dat de mechanismen die achter de verzending zitten te zichtbaar worden voor de gebruiker, vooral in de variërende vertragingen en foutmeldingen en hierdoor werd e-mail een springplank naar meer geavanceerde groupware systemen.
Distribution lists zijn *gelabelde* lijsten waarnaar een gebruiker een bericht kan sturen naar mensen van wie hij zelf bepaalt of ze dit ontvangen of niet. Een voorbeeld hiervan is het concept van ‘group’ in het adresboek van ‘Outlook Express’ [OE] (zie figuur 2.3-1). *Newsgroups* daarentegen zijn een manier om berichten te laten versturen naar een voor de verzender onbekend aantal onbekende ontvangers.



Figuur 2.3-1 Groep in Adresboek van Outlook Express

Bulletin boards en *forums* zijn een manier om te communiceren met honderden mensen. Het idee is dat men een onderwerp aanmaakt, dit kan een vraag, een mening of eender wat zijn waarop anderen kunnen repliceren en mensen kunnen daarop dan een antwoord sturen. Zo vormt er zich een boomstructuur waar mensen gemakkelijk informatie kunnen vinden en delen. Deze applicaties werken asynchroon, wat ervoor zorgt dat mensen met een niet-permanente internetverbinding er gemakkelijk gebruik van kunnen maken. Men hoeft immers niet te wachten tot de andere partij *"online"* is om een communicatie te starten. Het nadeel is dat men niet kan rekenen op een snelle respons en geen feedback van die respons heeft.

Synchrone tekstgebaseerde communicatie

In LAN-netwerken is de snelheid van verzending hoog genoeg om een conversatie te starten met e-mail. Een variant hierop zijn de UNIX 'talk' [Talk] en de VAX 'phone' [Vax]. Hier communiceren de gebruikers door tekst in te typen in een helft van het scherm, terwijl de tekst van de andere gebruiker wordt getoond op de andere helft. Hier wordt de tekst vaak karakter per karakter doorgezonden. Een variant hierop is de versie waar er slechts één transcript is. De granulariteit ligt hier lager, vaak wordt er slechts één lijn doorgezonden. De reden hiervoor is duidelijk, als de gesprekspartner het *transcript* venster volschrijft terwijl men nog aan het typen is, ziet men zijn eigen tekst niet meer. Voorbeelden hiervan zijn 'MSN Messenger' [Messenger] (zie figuur 2.3-2) en de vele *chat*programma's. 'ICQ' [ICQ], is een pakket waarmee er zowel synchroon als asynchroon kan gecommuniceerd worden.

Figuur 2.3-2 MSN Messenger

Structured Message Systems

Een groot probleem met e-mail en elektronische conferentiesystemen is het feit dat er een *overload* kan zijn voor de ontvangers. Stel dat iemand iets vraagt op een nieuwsgroep, dan wordt het al snel onoverzichtelijk om tussen alle ontvangen mails diegene te onderscheiden die betrekking hebben op die ene vraag. Gelukkig zijn er vaak meer passieve dan actieve gebruikers. Om deze problemen op te lossen werden er Structured Message Systems ontwikkeld. Een bekend voorbeeld hiervan is 'Information Lens' [InfLens]. Hier moet de verzender buiten 'Naar', 'Van' en 'Onderwerp' ook het type van zijn bericht kiezen, bijvoorbeeld een verwittiging voor een meeting. Het systeem zal dan een *template* of patroon voor dit bericht voorstellen waar hij, in dit geval, de tijd, plaats en titel zal moeten invullen. Gebruikers kunnen ook *filter agents* instellen, die met behulp van regels bepaalde acties verrichten als aan deze regels wordt voldaan.
Het probleem met deze systemen is dat de ontvanger wel meer voordelen heeft maar de verzender moet plots veel meer werk verrichten. Dit kan dan wel opgelost worden door het gebruik van standaarden en menu's met alternatieven voor de velden. Er is bovendien een debat tussen de mensen die vinden dat de gebruiker zijn structuren moet kunnen aanmaken, en die dat vinden dat dit door het systeem moet vastgelegd zijn. Dit heet het conflict tussen globaal – door de designers – en lokaal – door de gebruikers – structureren en komt nog in vele onderdelen van CSCW voor.

Video conferencing en communicatie

Het idee van beeld bij het telefoneren speelt al een tijd in de hoofden van de mensen, denk maar aan 'James Bond' en 'The Thunderbirds', en deze films bestaan toch al een kleine twintig jaar. Het probleem was dat er voor het gebruik ervan zeer dure kabels nodig waren maar met de introductie van recente en geavanceerde videocompressie technieken en ISDN en breedbandinternet in het algemeen zijn er reeds applicaties op de markt. De meest bekende is ongetwijfeld 'Netmeeting' [Netmeeting], en onlangs werd dit zelfs ook ingebouwd in 'MSN Messenger' [Messenger].
Er zijn drie wijdverspreide soorten van videogebruik: videoconferenties, video voor het verbeteren van sociale communicatie en video geïntegreerd in een andere applicatie. Deze zijn allen synchrone en remote faciliteiten.
Wanneer we videocommunicatie bekijken, zijn er volgens Fish et Al. [Fish et Al.92] drie belangrijke aspecten: toegankelijkheid, de mogelijkheid voor mensen om gemakkelijke toegang te hebben tot anderen, privacy, de mogelijkheid om te controleren welke informatie van zichzelf toegankelijk is voor anderen, en eenzaamheid, de mogelijkheid om inbreuk van anderen in iemands ruimte of het gebruik van iemands tijd te controleren.

Videoconferenties behoren op een vreemde manier tot de CSCW daar ze vaak geen gebruik maken van computers. De grens tussen telecommunicatie en computers wordt zo vaag dat het vaak als een gepast onderdeel van CSCW wordt beschouwd.
Hoewel de kwaliteit van videoconferenties nog een stuk lager ligt dan die van een normale meeting, wordt dit vaak verwaarloosd wanneer men de kosten van bijvoorbeeld het overvliegen van kaderleden bekijkt. Het probleem is de grootte en de kwaliteit van de beelden, lichaamstaal o.a. is moeilijk vast te leggen. Ook de positie van de camera is van cruciaal belang. Naar elkaar kijken is virtueel onmogelijk, aangezien de camera niet in het centrum van het scherm kan staan. Om het gevoel te geven van een sociale aanwezigheid bestaan er 'videomuren', waar een levensgroot beeld van de andere partij wordt getoond. Kosten hiervoor zijn natuurlijk enorm, privacy wordt mogelijk geschonden in die gedeelde ruimte. Uit het zicht van – of zelfs te dicht bij – de camera lopen is ook een mankement.

Virtuele collaboratieve omgevingen

Met de opkomst van 3D technieken is ook de *virtual reality* ontstaan en het werd dus een kleine stap om er meetings of interacties tussen gebruikers in het algemeen in te organiseren. Elke gebruiker krijgt via zijn systeem een virtuele wereld voorgeschoteld, zij het via een desktop applicatie of via *immersive VR*, terwijl anderen hun zien als een kubusgebaseerde figuur, al dan niet met een foto of zelfs videobeeld erop afgebeeld. Zo'n objecten noemen we '*embodiments'* of '*avatars*' als men het over concept van virtuele personen in 't algemeen heeft en wanneer de gebruikers rond bewegen in de virtuele omgeving, bewegen die embodiments respectievelijk. Zo kunnen er gesprekken ontstaan, kan men gezamenlijk objecten bespreken of kan men zelfs bestanden meenemen in die virtuele ruimte, zodat men bijvoorbeeld samen een tekst kan herzien. Het probleem hiermee is de leesbaarheid van de tekst, en de '*shared focus*'. Als één embodiment ergens naar wijst, moet dit bij de andere gebruiker(s) op hetzelfde stuk tekst wijzen.
Om de omgeving zo dicht mogelijk bij de echte wereld te laten staan is er een goede geluids*rendering* nodig, als iemand ver van je is in die omgeving, mag je hem niet of zeer stil horen, anders hoor je een chaos van geluiden. Ook de plaats waar het geluid vandaan komt moet goed worden weergegeven.
Omgevingen als deze worden steeds meer en meer gebuikt voor militaire en industriële training in team, collaboratief ontwerpen en *multiplayer* spellen. Toekomstige commerciële applicaties omvatten virtuele winkelcentrums en showrooms, conferenties, klantenondersteuning en leren op afstand.
Hoewel deze werelden nog grotendeels in de research fase staan, zijn er reeds 2D virtuele werelden beschikbaar op het Web. Dit zijn MUD's of *Multi User Domains*. Hier wordt men voorgesteld als een beeldje of een icoon, en zo kan men ook andere mensen tegenkomen en tekst- of zelfs audiogebaseerde

conversaties starten met de andere gebruikers. Een bekend voorbeeld hiervan is 'The Edge of Creation' [EdgeOfCreation].

2.3.2 Meeting en decision support systemen

De bedoeling van deze systemen is het genereren van ideeën en een gelijke verstandhouding. Dit moet voornamelijk gebeuren in onderzoeksomgevingen, tijdens ontwerptaken, in management vergaderingen en in brainstorming sessies. Er zijn drie grote soorten systemen die behoren tot dit onderdeel, en dit zijn de argumentatie *tools*, die argumenten die gebruikt worden om tot een beslissing te komen moeten opslaan, elektronische vergaderruimtes, die *real-time* vergaderingen ondersteunen en tenslotte *shared drawing surfaces*, gebruikt voor de ondersteuning van design vergaderingen.

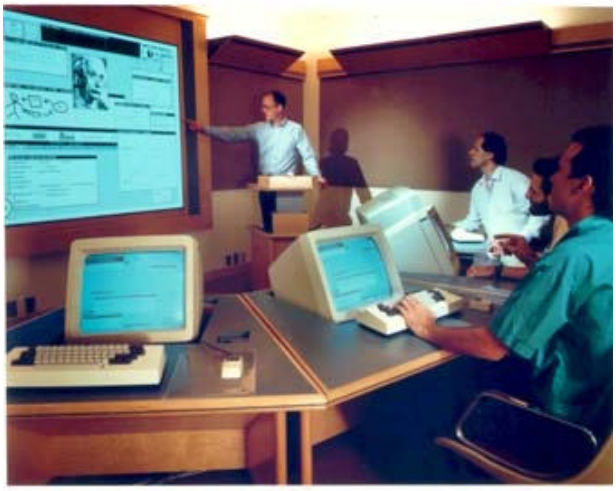
Argumentatie tools

Vaak kunnen we beslissingen bij het ontwerpproces opslaan in commentaar die erbij gevoegd wordt. Hierbij denk ik aan 'Rational Rose' [Rose] waar er bij elke klasse en bij elk object in het algemeen een notitie kan gevoegd worden. Dit is asynchrone communicatie naar latere gebruikers van het ontwerp toe. Nu bestaan er ook hulpmiddelen die dit ondersteunen tussen verschillende ontwerpers simultaan. Vaak hebben ze een soort hypertext structuur, en kunnen ze gemakkelijk ontwerperteams als individuele designers ondersteunen. Als verschillende mensen samen werken, moeten er mechanismen zijn om te verhinderen dat hun werk interfereert. Dit noemen we *concurrency control*. Bij hypertext is dit gemakkelijk, er moet eenvoudigweg voor gezorgd worden dat verschillende mensen niet op dezelfde knopen werken, en hiervoor worden knopen die bezet zijn geblokkeerd, wanneer één gebruiker aan een *node* werkt, kan een andere dezelfde niet aanpassen. Een bekend model ter ondersteuning van argumentatie is 'IBIS' [IBIS].
Argumentation tools zijn grotendeels asynchroon co-located, hetzelfde systeem wordt om beurten door verschillende mensen gebruikt maar zoals boven beschreven is er geen belemmering om het ook remote en synchroon te gebruiken.

Elektronische vergaderruimtes

Men kan zich afvragen waarom er nood is aan dure elektronische vergaderruimtes, alle leden van de vergadering zitten namelijk samen, dus waarom dure computerapparatuur plaatsen in speciaal gebouwde ruimtes? Toch bestaan er een tiental zo'n kamers, waaronder Xerox PARC Colab

[Colab] (zie figuur 2.3-3), die tamelijk bekend is. Meestal zitten er in zo'n kamer een aantal mensen, elk met hun eigen scherm, in een U-vorm rond een centraal scherm, dat dienst doet als elektronisch *whiteboard*. Vaak tonen de schermen van alle participanten alsook het centrale scherm hetzelfde beeld, wat bekend staat als WYSIWIS (*What You See Is What I See).* Voordelen van zo'n bord zijn dat men er niet alleen kan op tekenen en afvegen maar ook onderdelen verslepen, opslaan en afprinten. Ook hier kan er een vorm van blokkeren aanwezig zijn, om ervoor te zorgen dat twee mensen niet op dezelfde plaats tekenen, hoewel dit niet altijd nodig is. Op het eigen scherm kan men immers zien dat je samen aan een tekening begint. Het probleem hier is dat moeilijk alle cursors kunnen getoond worden; in sommige systemen kunnen er namelijk tot dertig mensen deelnemen. Vaak worden cursors dan weggelaten, wat bovendien een extra aspect van anonimiteit geeft, en wat van pas kan komen tijdens brainstorming fases. Hier rijst echter een nieuw probleem op: er kan geen gebruik gemaakt worden van *deixis*. Dit betekent dat we niet naar het bord kunnen verwijzen met de cursor om andere mensen duidelijk te maken waarover we bijvoorbeeld een opmerking hebben. Maar ook hier kan er een oplossing voor gevonden worden: een groep aanwijzer. Dit is een icoontje dat op elk scherm wordt getoond en waarvoor men ook het gebruik moet aanvragen.



Figuur 2.3-3 XeroxPARC Colab

Shared work surfaces

Voorgaand concept kan men natuurlijk ook toepassen op remote wijze, om zo van op verschillende plaatsen op een gezamenlijk whiteboard te werken. Bijkomende problemen zijn dan dat men minstens een geluidsverbinding moet hebben, en over een video beschikken is ook geen overbodige luxe. Het zou immers te onhandig zijn om samen te werken, als men enkel op het bord moet schrijven. Waar het in real-time vergaderingen gemakkelijk op te merken is dat twee mensen met de gezamenlijke aanwijzer werken, gewoonlijk is de persoon die met de aanwijzer bezig is ook de spreker, is dit hier minder eenvoudig op te merken. Dit omwille van minder communicatie tussen de participanten en vertragingen.
Computernetwerken zullen meer moeite hebben om de informatie te slikken, zeker als men er geluid en video bij wil gebruiken. Netwerkvertragingen zullen het dan ook moeilijker maken om de gebruiker te laten ontdekken wanneer twee mensen op dezelfde plaats schrijven, dus concurrency control mechanismen zullen iets minder toegeeflijk moeten zijn.
Sommige systemen laten de gebruikers rechtstreeks op een scherm tekenen, en geven aan de anderen een beeld van hun schrijven, handen of zelfs lichamen weer, door middel van camera's en spiegels. Dit om een meer realistisch beeld weer te geven en gebruik te laten maken van lichaamstaal.

2.3.3 Gedeelde applicaties en artefacten

In dit onderdeel behandelen we het soort van groupware waar het samenwerken geen ondersteuning biedt aan een of ander proces (bijvoorbeeld een meeting of designproces) maar waar de applicatie, de computer of een document gedeeld wordt.

Gedeelde Pc's en vensters

Het delen van Pc's en vensters zorgt ervoor dat gewone applicaties het onderwerp kunnen zijn van collaboratief werk. De bedoeling van gedeelde Pc's is dat meerdere computers samenwerken alsof ze één zijn, terwijl bij gedeelde vensters systemen het de vensters zijn die gedeeld worden. Het verschil met eerder besproken *shared work surfaces* is dus dat we met gewone (niet-collaboratieve) applicaties werken, en wat die applicatie betreft is er dus maar één muis en keyboard. Het eerste probleem dat we opmerken is dat twee gebruikers samen een woord op dezelfde plaats willen typen, en als gevolg om beurten letters van een woord schrijven. Hier hebben we dus een vorm van locking nodig. Voor de muis zal dit een automatisch *lock* zijn, dat van toepassing is wanneer de muis bewogen wordt, en zeer snel – bij een

moment van inactiviteit – opnieuw losgelaten wordt. Bij het toetsenbord zal die periode van inactiviteit iets langer moeten zijn.
Zoals de elektronische vergaderruimtes kan deze software zowal lokaal als op afstand gebruikt worden, dan weer best met audio- en videokanalen, aangezien die software geen andere directe communicatie biedt. Men kan natuurlijk wel bijvoorbeeld berichtjes schrijven in de *workspace* van de applicatie.
Een veelgebruikt domeinen waar men dit vindt is in het oplossen van technische problemen – Windows XP [WindowsXP] biedt nu bijvoorbeeld een 'Hulp op afstand'-faciliteit aan – en het gezamenlijk aanmaken van documenten, bijvoorbeeld van een *spreadsheet*.

Gedeelde verwerkers

Een gedeelde tekst- of beeldverwerker (of een ander type gegevens) is een hulpmiddel voor het monteren en verbeteren van tekst of een andere soort elementen dat speciaal ontworpen is om dat verschillende personen gelijktijdig gebruikt te worden. Niet dus zoals in voorafgaande paragraaf, waar een speciale applicatie alles verzorgt. Het voordeel is dat de blokkeringprotocollen speciaal afgestemd kunnen worden op het gedrag van de verwerker zelf. Om efficiënt te kunnen werken is er meestal een supplementaire vorm van communicatie nodig, al was het maar onder de vorm van tekst.
Er zijn een aantal design opties waarmee rekening gehouden moet worden, bijvoorbeeld of er één enkel insertiepunt moet zijn of zou de gebruiker eender waar zijn tekst kunnen intypen? Zien alle gebruikers hetzelfde deel van de tekst, hebben ze dezelfde weergave ervan (afdrukweergave, overzichtsweergave, …) en als er iemand, in de pagina schuift, verandert het zicht bij de anderen ook?
Veel hangt af van hoe groot de granulariteit is, als de gebruikers samen aan een zin kunnen werken heeft het bijvoorbeeld al meer zin om een WYSIWIS te hebben dan als ze slechts per paragraaf kunnen aanpassen. WYSIWIS is dus handig als er sprake is van 'echte' synchrone collaboratie, bijvoorbeeld als ze samen een zin bekijken weten ze dat de zin in het midden van de pagina ook dezelfde bij de andere gebruiker is. Maar er stellen zich dan nieuwe problemen. Er zijn natuurlijk de *typing-clashes* maar bovendien komen er problemen met de schuifbalken ('*scroll wars*') bij, en die zijn nog moeilijker op te lossen, door de minder duidelijke *feedback* en voorspelbaarheid. Er moet dus nagedacht worden over een goed blokkeringmechanisme, over een gedeelde muiswijzer

Co-authoring systemen

Het verschil met het gezamenlijk verwerken van vorige paragraaf is dat men hier geen tekst gaat verbeteren en bewerken maar samen tekst gaat opstellen. Terwijl *editing* een paar uren duurt, gaat *authoring* over een paar weken of maanden. De kans dat de gebruikers dus samen aan een zelfde tekst stukje werken is dus veel kleiner, en deze actie is dus grotendeels asynchroon. Gezamenlijk tekst verwerken kan hier een onderdeel van zijn of gebruikers kunnen eenvoudigweg kladversies doorgeven en zo mekaars werk becommentariëren. Wat we kunnen besluiten, na veelvuldige studies over individueel en coöperatief schrijven is dit: iedereen en elke groep is verschillend.
De basisstructuur van het document kunnen we wel vastleggen en vaak wordt er *hypertekst* gebruikt. De vorm hiervan is een *graph* of gelimiteerd tot een boom, waarvan de knopen geblokkeerd kunnen worden of waar er zelfs synchrone activiteit kan plaatsvinden. Dit is vaak niet de gebruikelijke manier bij auteursystemen, per sectie is er een hoofdauteur, en onderlinge afspraken zullen schrijfconflicten vermijden.
'Quilt' [Quilt] is een voorbeeld van zo'n *co-authoring* systeem, waar gebruikers *rollen* hebben, zoals auteur, commentator of lezer. Een auteur kan een tekst aanpassen, een commentator kan notities maken en een lezer kan alleen lezen. Hier komt het aspect van locale ten opzichte van globale structurering weer aan bod, in de echte wereld kunnen die rollen flexibel aangepast worden, in groupware zou dit moeten kunnen bijgehouden worden.

Gedeelde agenda's

De reden waarom deze applicaties zijn ontstaan is eenvoudig: vaak moeten verschillende personen afspraken maken en er moeten dus gezamenlijke tijdstippen gevonden worden in de agenda's van de verschillende personen. Dit is natuurlijk handiger en sneller te verzorgen via een computer systeem dan manueel. Het probleem dat zich hierbij stelt is dat het systeem een oplossing moet vinden als er geen gemeenschappelijke vrije periodes zijn. Oplossingen zijn: een aantal afspraken suggereren om te verplaatsen; werken met belangrijkheidniveaus, zodat de kost voor elke afspraakbreuk kan berekend worden. De gebruiker zal echter een belangrijke rol blijven spelen, hij moet immers op de hoogte gehouden worden en soms ultieme beslissingen zelf nemen, bijvoorbeeld bij conflicten die niet aan de hand van eerder besproken belangrijkheidniveaus kunnen worden opgelost.
Een ander aspect waarmee rekening moet gehouden worden is de privacy: sommige mensen hebben geen recht om bepaalde zaken in je agenda te zien, privé versus zakelijk bijvoorbeeld Dit wordt dus ook het best in het systeem ingebouwd.

Er zijn ook mogelijke problemen met het aanvullen van agenda's door derden. Mag iemand zomaar iets invullen in een leeg periode? Vaak is er in zulke elektronische agenda's mogelijkheid om potloodaantekeningen te maken, die moeten bevestigd worden door de eigenaar.

2.4 Implementeren van synchrone groupware

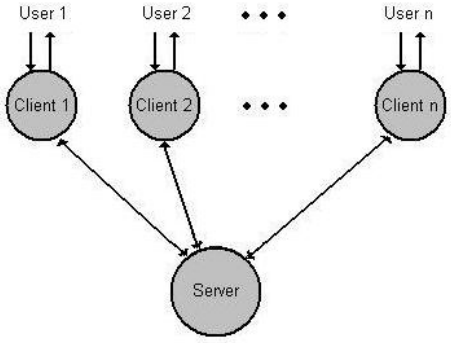
In dit deel gaan we bekijken welke problemen optreden bij het implementeren van groupware, voornamelijk bij synchrone. Het spreekt voor zich dat het bouwen ervan moeilijker is dan dat van systemen voor een enkele gebruiker. We moeten immers ervoor zorgen dat de verschillende aanpassingen die we krijgen van de gebruikers niet resulteren in een warboel op hun schermen, bovendien wordt dit bemoeilijkt door de netwerkvertragingen.

2.4.1 Feedback en netwerk vertragingen

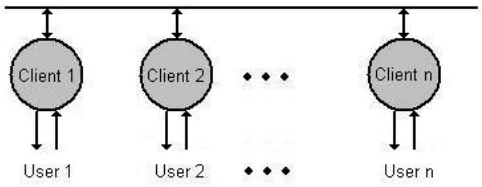
Feedback; het verkrijgen van een respons op wat er werd gedaan, hangt af van de applicatie waarmee de gebruiker werkt. Bij editing software is het belangrijk dat bij een aanpassing van een karakter dit ogenblikkelijk op het scherm verschijnt, bij het ingeven van tekst mag dit al iets langer duren, terwijl het bij tekenen nog korter moet zijn dan bij editing. We kunnen voor deze respons dus moeilijk een netwerkoperatie uitvoeren, aangezien dit te lang zou duren. Er zouden minimaal twee netwerkberichtjes zijn, als er dus geen verloren gaan en geen *handshakes* zijn. Bovendien gaat dit vaak over meer dan één computer met *server* maar soms een tiental stations. Netwerkvertragingen moeten in deze architecturen dus ernstig genomen worden.

2.4.2 Architecturen voor groupware

Er zijn twee grote soorten architecturen voor groupware, namelijk gecentraliseerd (of client/server) en gerepliceerd (vele kopietjes van dezelfde basisstructuur) met variaties erop. In een gecentraliseerd systeem heeft elke deelnemer een minimaal programma of een client die het beeldscherm en de invoer van de gebruiker voor zijn rekening neemt. De echte bewerkingen gebeuren aan de server kant, deze houdt de data bij op een centrale, vaak krachtige computer (zie 2.4-2). Deze systemen zijn het meest eenvoudig te implementeren. Een speciale vorm hiervan is de *master-slave* architectuur, waar er een server onzichtbaar draait op de computer van de eerste persoon die de applicatie opstartte.



Figuur 2.4-2: Client/server architectuur ([Dix et Al. 98])



Figuur 2.4-1: Gerepliceerde architectuur ([Dix et Al. 98])

De tweede vorm is gerepliceerd (zie 2.4-1), waar iedere gebruiker een kopie van de applicatie draait die allen met elkaar communiceren en de datastructuren consistent met elkaar proberen te houden. De bedoeling hiervan is de impressie van een gecentraliseerde applicatie te geven maar de prestaties van een gedistribueerd systeem te bereiken.

Vergeleken met een client/server architectuur is een gerepliceerd systeem moeilijker te programmeren. Het probleem is natuurlijk de data overal gelijk te houden, terwijl op twee applicaties de data gelijktijdig aangepast kan worden door verschillende mensen. Dit is een veelvoorkomend probleem, en de meest voorkomende oplossing is een *roll-back* op één of andere replica, waar de oorspronkelijke staat hersteld wordt en de operatie opnieuw uitgevoerd wordt. Dit kan natuurlijk niet als de resultaten reeds werden getoond op één van de schermen, dit is een standaard algoritme. Die zijn vaak niet van toepassing op groupware. Door middel van blokkeringmechanismen of *floor control mechanisms* worden zulke problemen vermeden. Standaard mechanismen kunnen gebruikt worden als *feedback* niet belangrijk is.
Het grote voordeel van een replica-architectuur is de snelle feedback maar ook bij een client-systeem kan er een deel van de respons door de clients zelf opgevangen worden. Vaak wordt de server enkel een opslagmedium, en wordt de functionaliteit grotendeels door de clients opgevangen. Anderzijds zal een gerepliceerde architectuur de clients zelden identiek behandelen, opslaan en ophalen zal niet op elke replica gelijktijdig gebeuren. Er is dus geen duidelijke scheiding tussen de twee soorten systemen.

2.4.3 Shared States

Dit is ook een belangrijk punt bij het creëren van gedeelde applicaties in het algemeen. We willen, bijvoorbeeld in een gedistribueerde virtuele omgeving, de toestand van de verschillende gebruikers delen. Dit is logisch, iedereen moet weten waar de andere participanten zich bevinden of wat ze ingevoerd hebben in hun systeem. Er zijn dan drie manieren om de gegevens te delen, namelijk de centrale opslagplaats (*centralised repository)*, het *dead-reckoning* algoritme en hiertussen de frequente toestandsgeneratie (*frequent state generation)*. Natuurlijk zijn dit niet de enige mogelijke methoden maar het zijn de drie voornaamste structuren, waarop variaties kunnen geïmplementeerd worden.

Centrale opslagplaats

De gedeelde data wordt centraal bewaard, en kan centraal gelezen en aangepast worden. Door middel van blokkeringmechanismen wordt ervoor gezorgd dat de gegevens consistent blijven. Om de snelheid te verbeteren (iedereen moet immers lezen en schrijven op één bepaalde plek, dus de *bottleneck* kan significant zijn), is er de mogelijkheid om gebruik te maken van een *cache*. Dit is een versie van de data bij elke participant afzonderlijk, de enige restrictie is dat degene die de gegevens aanpast er wel moet voor zorgen dat de cache bij alle participanten aangepast wordt.
Mogelijke knelpunten bij deze methode zijn de bandbreedte- en ook de prestatie-bottleneck, één systeem moet immers alle gegevens bijhouden en aanpassen. De fouttolerantie is een ander mogelijk probleem. Als het centraal systeem uitvalt kan geen enkele participant nog werken. Ook kunnen we niet voorspellen hoelang het duurt om over bepaalde gegevens te kunnen beschikken, en er is een significantie communicatie *overhead*.
Dit model biedt natuurlijk ook voordelen. Het is een eenvoudig model, dat gemakkelijk te implementeren en te begrijpen is. De informatie zal ten allen tijde consistent blijven, en er kan geen dispuut zijn over de eigendom van de data. Waarom eigendomsrechten belangrijk zijn zien we onder sectie 2.4.2, in ‘Frequente toestandgeneratie’.

Frequente toestandgeneratie

Dit algoritme gebruikt men als men absolute consistentie niet echt nastreeft, bijvoorbeeld als er veel opeenvolgende aanpassingen moeten gebeuren op dezelfde gegevens. De bedoeling is dat elke wijziging naar iedereen wordt gestuurd, via een blinde broadcast, zonder dat de anderen hierop een bevestiging moeten terugzenden. Bovendien is er geen globale ordening van de pakketjes, zodat de lengte en de verwerking ervan kort gehouden kunnen worden. Gevolg is dat een applicatie twee pakketjes in verkeerde volgorde

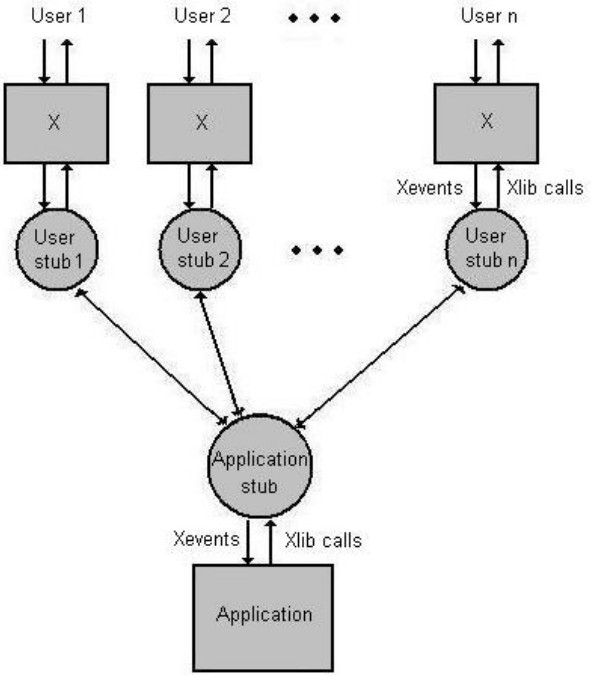
kan krijgen en zodoende een wijziging aanbrengt die de toestand achteruit zet, zonder dat dit een echt probleem is (of zou mogen zijn). Stel bijvoorbeeld dat twee gebruikers bijna tegelijkertijd een schuifbalk hanteren, dan kan het gebeuren dat de twee Pc's tegengestelde informatie naar de anderen zenden, en dat de ene dus een totaal andere positie op de schuifbalk toont dan de andere. Dit kan vermeden worden door een eigendomsrecht toe te kennen aan die schuifbalk of eender welk gedeeld object. Als men dit object dan wil manipuleren moet men ofwel het eigendomsrecht verkrijgen ofwel het laten doen door middel van een zogenaamde *proxy update*, de proxy zijnde de eigenaar, die zelf zijn schuifbalk in opdracht van iemand anders beweegt.
Knelpunten bij deze methode zijn niet alleen de bandbreedte die voldoende moet zijn maar ook de latentie. Er moeten zoals reeds gezegd veel berichtjes worden doorgestuurd, en met een trage of te veel variërende overdrachtsnelheid kan men aan redelijk grote verschillen tussen de participanten onderling komen. Maar zelfs al zijn de bandbreedte en latentie ruimschoots bevredigend dan mag men zich toch nog niet aan volledige consistentie verwachten.

Dead reckoning

Bij dit type systeem gaat men een minimale bandbreedte gebruiken, als gevolg zal de consistentie tussen de participanten onderling niet heel groot zijn. De bedoeling is dat de participanten af en toe (bij significante wijzigingen) posities doorsturen maar dat op elk systeem aan de hand van voorspellingen de toestand van de objecten wordt getoond. Dit model wordt dus vaker gebruikt bij spelletjes en simulaties. Als bijvoorbeeld een berekende baan dan afwijkt van de volgende positie die toegezonden werd, kan de avatar verspringen naar de juiste positie of ernaar convergeren via een algoritme. Dit gebeurt wel vaker gezien de beperkte informatie. De term *dead reckoning* komt uit het principe dat men niets verandert aan de positie of zelfs aan de baan (indien het voorwerp in beweging is) als er geen berichtjes meer binnenkomen, tot er een bepaalde tijdspanne om is. Dan gaat men zelf informeren bij de verzender, om bijvoorbeeld na te gaan of er geen informatie uit pakketjes verloren gegaan is of om te checken of een participant nog aanwezig is.
Waar men hier dus mee rekening mee moet houden is het feit dat er geen consistentie is tussen de systemen onderling; ze kunnen zelfs veel verschillen, en in geval van simulaties kan dit een probleem zijn. Collisie detectie, wat heel belangrijk kan zijn in een legersimulatie, zal dan niet voldoende accuraat zijn. Geraakt worden door een kogel of die ontwijken kan een enorm verschil maken in de uitslag van die simulatie! Ook hebben we, per systeem, genoeg kracht en geheugen nodig om alles te berekenen. Wel kunnen we met dit systeem veel participanten laten samenwerken.

2.4.4 Shared window architectures

Dit zijn de applicaties die gemaakt werden voor één gebruiker maar door een programma als groupware gebruikt worden. Ze hebben enkele gelijkenissen met normale groupware maar ook enkele verschillen. In een gewone applicatie zal de gebruiker met de *window manager* communiceren, zodat deze manager de input doorgeeft aan de applicatie, en omgekeerd. Bij een *shared window architecture* zal er een applicatie*stub* de feedback opvangen en naar de *user stubs* doorsturen, en omgekeerd (Zie figuur 2.4-3). De invoerkant heeft een floor control nodig, vooral voor de muis. Dit kan de applicatiestub voor zijn rekening nemen, bijvoorbeeld door te kijken naar speciale karaktercombinaties. De gebruikerstub kan andere taken toevoegen, zoals nieuwe stukken *user interface*, een knop om de floor aan te vragen, een overzicht van de andere gebruikers…

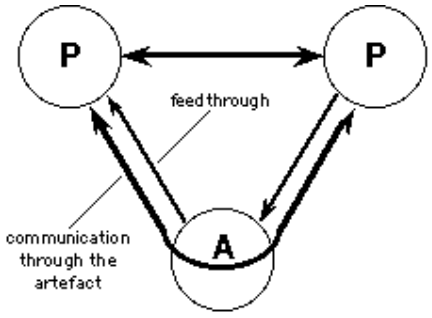


Figuur 2.4-3 Shared window architecture ([Dix et Al. 98])

Het grootste probleem met de onderliggende applicaties is dat ze vaak niet geschreven werden om gedistribueerd gebruikt te worden en dat er vaak grote datastructuren aan te pas komen. Vaak wordt er hier daarom een master-slave architectuur toegepast.

2.4.5 Feed through en netwerkverkeer

Buiten de feedback die meestal zo snel mogelijk moet doorgegeven worden aan de gebruiker, is er ook het aspect van *feed through*, het doorsturen van informatie naar de andere participanten. De vereisten liggen hier niet zo hoog als bij feedback, dit wil zeggen dat meestal het niet zo snel moet getoond worden. Een belangrijk aspect bij feed through is dat de communicatie door het artefact gaat, en niet rechtstreeks van de andere gebruikers komt. Ook wanneer we in het echte leven samenwerken met fysieke objecten, is de communicatie door het artefact vaak belangrijker dan directe communicatie. Een gebruiker kan je altijd meedelen dat hij een aanpassing heeft uitgevoerd maar zolang ze niet langs het systeem wordt doorgepropageerd heb je geen bewijs dat hij dit daadwerkelijk deed of dat ze effectief is opgenomen in dit systeem.



Figuur 2.4-4: Feed Through ([Dix96])

Stel dat een gebruiker een verandering aanbrengt, bijvoorbeeld een nieuw karakter typt. Als dit dan één voor één moet doorgezonden worden naar de andere gebruikers, (stel dat er *n* zijn), dan moeten er ook *n* berichtjes verzonden worden. Telkens als er een muis wordt verplaatst is een berichtje verzenden dus niet mogelijk. Daarom zijn er verschillende oplossingen bedacht, zoals het *broadcasten* van berichtjes, wat evenwel niet altijd wordt ondersteund, en het verhogen van de granulariteit van berichtjes. Dit wil zeggen dat men de berichtjes groter maakt, er meer gegevens in stopt en er dus minder stuurt. De grootte van berichtjes moet ook afgewogen worden met de aard van het programma. Een derde oplossing is de applicaties in een ring plaatsen om ze zo berichtjes naar elkaar te laten doorsturen. Niet iedereen zal dan tegelijkertijd het bericht ontvangen maar er zal niet zo'n grote wachtrij aan berichtjes zijn op één participant als wanneer elke applicatie zijn veranderingen naar alle andere moet sturen.

2.4.6 Grafische toolkits

Het gebruik van *widgets*, zoals *menus*, knoppen, *dialog boxes*, … door middel van bestaande grafische toolkits kan ook een probleem vormen bij de implementatie van groupware. In een *text area* kunnen we bijvoorbeeld slechts één cursor weergeven, en nuttige informatie, zoals welk deel van de tekst nu door de gebruiker wordt bekeken, kan er ook niet opgevraagd worden. Vaak moet iemand dus de applicatie ontwerpen rond de bestaande widgets of zelf zaken implementeren.

2.4.7 Robuustheid en schaalbaarheid

Er zijn vier soorten problemen die kunnen optreden in deze context:

- netwerk, *operating systems* of *workstations* die slecht werken..
- programmeerfouten in de gedeelde applicatie
- onvoorziene opeenvolging van gebeurtenissen
- het systeem kan een groot aantal gebruikers niet aan

De eerste drie kunnen ook voorkomen in niet collaboratieve systemen maar groupware maakt de kans op deze problemen alleen maar groter: er is immers een groter aantal verschillende software; de algoritmen zijn complexer: door het grotere aantal verschillende gebruikers is het zich voordoen van onvoorziene opeenvolgingen van gebeurtenissen groter. Het ontwikkelen en testen gebeurt bovendien vaak op één station, zodat er vaak nieuwe problemen opduiken wanneer het systeem verspreid wordt op diverse toestellen.

Server fouten

Om een servercrash tegen te gaan, wat toch de meest voorkomende fout is van serverzijde, moet er geregeld automatisch opgeslagen worden. De datum wordt als laatste in zo'n file opgeslagen, zodat, als die er niet staat, de crash gebeurd is tijdens zo'n save en dit ook ontdekt kan worden.

Workstation fouten

Deze problemen gebeuren frequenter dan die met servers, door de meer ingewikkelde software en het groter aantal hiervan. Het doel is van de crash tot één gebruiker te beperken en zo snel mogelijk te herstellen.
De server moet robuust zijn, het falen van een client mag het hele systeem niet doen hangen, doordat hij bijvoorbeeld op een respons wacht. Hij moet ook zien dat een client ontbreekt, door middel van een *timeout* bijvoorbeeld en moet de andere clients hiervan op de hoogte brengen. Tenslotte moet hij aan die client de kans geven zo snel mogelijk te herstarten.

Hetzelfde geldt bij replicastructuren, waar de andere moeten zorgen voor het ontdekken van het ontbreken van één van de participanten.

Algoritme fouten

Sommige fouten doen de applicatie niet crashen en hierdoor kunnen ze moeilijker te ontdekken zijn. Dit kan als gevolg hebben dat datastructuren tussen verschillende deelnemers inconsequent zijn. Kritische algoritmes moeten intensief getest worden, en het is vaak niet overbodig om tussen de code controles te stoppen, hoewel dit programmeer- en executietijd kan kosten. Een minimale vereiste is een *reset*, waar alle deelnemerprogramma's gesynchroniseerd worden, terwijl enkel de meest recente aanpassingen verloren gaan.

Onvoorziene sequenties van gebeurtenissen

Deadlock is het meest bekende probleem in gedistribueerde systemen, waar twee clients op mekaars antwoord wachten. Daarom moeten er nooit operaties gebeuren die het systeem kunnen laten wachten. Er mogen geen aannames gemaakt worden over de volgorde van inkomende gebeurtenissen. Ook de aanname dat berichten verzonden van één computer onder dezelfde vorm aankomen in de andere kan een probleem stellen. De lengte kan immers verschillen, dus de lengte in de *header* schrijven kan hiervoor een oplossing betekenen.

Scaling up

Problemen kunnen zich ook voordoen wanneer het aantal gebruikers toeneemt. Sommige datastructuren, zoals tabellen en algoritmen, voldoen niet voor meerdere gebruikers. Hiermee zal moeten rekening gehouden worden in het begin. Het aantal berichtjes, het openen van bestanden en netwerkconnecties kan beperkt zijn door het besturingsysteem.

3 Het Internet

3.1 Introductie

Om op wetenschappelijk gebied degelijke vooruitgang te boeken is het nodig om verder te gaan op bevindingen van andere mensen. Bovendien groeit het aanbod van literatuur dagelijks. De nood aan een algemene verzamelplaats voor informatie, vóór het ontstaan van Internet, was reeds geruime tijd aanwezig. In 1945 ontwierp Vanevar Bush in zijn bekende werk 'As we may Think' het prototype voor het Internet, de Memex [Memex]. Het was een futuristisch systeem om informatie te bekomen, onder de vorm van fotokopieën van gedocumenteerde informatie, met links tussen de delen van documenten onderling. Het systeem had enorm veel gelijkenissen met computers en het Internet nu, behalve dat het om informatie op microfilm ging i.p.v. digitale informatie.

Midden de jaren '60 creëerde Ted Nelson de Hypertext [Hypertext], een niet-lineaire tekststructuur en midden jaren '80 ontstonden de eerste hypertext systemen.

Het Internet, gegroeid uit het ARPA netwerk, ontstond als een verbinding tussen vier universiteiten. Langzaamaan breidde het uit, en in 1982 werd de TCP/IP erop toegepast. In 1987 bereikte het de kaap van 10.000 *host* computers, waarna het een steile klim kende. In 1990 ontstond de term *World Wide Web*, en in '93 waren er reeds 3 miljoen *host* computers.

In eerste instantie was het Internet gebouwd als een onderzoeksomgeving voor het delen van informatie. Er was dus geen nood aan een afdoende beveiliging, aan mooie presentatie en aan programmeerfaciliteiten. Met de ontwikkeling van 'Mosaic' [Mosaic], de eerste *webbrowser*, kwam daar stilaan verandering in. De eerste generatie, die van 1993, was ontwikkeld om tekst met weinig beelden te tonen, zonder lay-out en andere visuele aantrekkelijkheden. Daar kwam al snel verandering in met de volgende generatie in '94. Bij web sites begon het meer rond presentatie dan rond inhoud te draaien en de eerste designers kwamen in beeld. Met de opkomst van Internet Explorer, concurrent van Netscape Navigator, was er geen stoppen aan de ontwikkeling van nieuwe technologieën en concepten, zoals *frames, style sheets, JavaScript, …*

Het World Wide Web wordt dus vaak bekeken als de nieuwste en meest gebruiksvriendelijke manier om informatie aan te bieden over het Internet.

Met de ontwikkeling van nieuwe technologieën werd het echter ook mogelijk om applicaties te ontwikkelen, en collaboratieve in het bijzonder, door middel van Java, JavaScript, CGI-scripts (hoewel er toch wat beperkingen zijn). Het Web is meer geworden dan een documentent server.

In de volgende paragrafen gaan we de voor- en nadelen van de WWW-technologieën bekijken in de context van de ontwikkeling van groupware. De voornaamste uitdagingen waarvoor deze en alle vormen van groupware staan, liggen op drie gebieden. Vooreerst zijn er de sociologische aspecten die te maken hebben met het goed ondersteunen van groepen. Dit omvat het goed begrijpen en het toegepast zijn op de interacties van de groep. Het nut dat de nieuwe applicatie biedt moet immers significant groter zijn dan de moeite die de gebruikers moeten doen om ze te leren gebruiken. Vervolgens zijn er de *UI-design* aspecten: de coördinatie van meerdere gebruikers en het verspreid gebruik van de software. Tenslotte zijn er de technische aspecten: vooreerst degene die te maken hebben met het netwerk, zoals algemene toegang en beveiliging, communicatie, privacy en synchronisatie; en dan die van de distributie, het inzetten en het onderhoud.

3.2 Het World Wide Web

Het ‘World Wide Web’ of het WWW is de bekendste manier om documenten te verspreiden over het Internet. Dit gebeurt onder de vorm van Hypermedia documenten, dit zijn documenten die bestaan uit Hypertext en multimedia. De inhoud van zo’n documenten bestaat vaak uit tekst, tekeningen en foto’s en soms ook uit video en geluid. Men kan van document naar andere of naar een andere plaats in hetzelfde document navigeren/springen door middel van links. Meer en meer wordt het WWW ook gebruikt om applicaties aan een breed publiek aan te bieden. Men werkt er met een gedistribueerde client/server architectuur. Met het WWW zijn er een aantal nieuwe standaarden opgedoken en er komen er geregeld bij. HTML, URL, HTTP, MIME en CGI zijn de basistechnologieën. Voor het programmeren van ingewikkeldere clients en servers zijn er o.a. JavaScript voor clients en ASP en CGI-scripts voor servers.

3.2.1 De basis standaarden

HTTP

Hypertext Transfer Protocol of HTTP [HTTP]

URL

Universal Resource Locators of URLs zorgen voor de adressering. Zij hangen af van 'Domain Name Services', het Web is immers onderverdeeld in domeinen die een naam krijgen van DNS servers.

MIME

Multipurpose Internet Mail Extentions of MIME, dit zijn extensies zoals .jpeg, die aanduiden welke informatie er eigenlijk wordt doorgestuurd.

3.2.2 Programmeren aan de client zijde

Programmeren aan de clientzijde is het ontwikkelen van applicatiecode die door de clients wordt gebruikt om informatie aan gebruikers te tonen, input van gebruikers op te vragen, hulpbronnen van de applicatie te initiëren, controleren en altereren en om lokaal gebruikerinteracties met de applicatie te verwerken. We bespreken achtereenvolgens kort de belangrijkste concepten en technologieën op dit gebied,

HTML

Hypertext Markup Language of HTML [HTML] is de *markup* taal waarin de documenten worden geschreven. Er is, dankzij Netscape [Netscape] in '95, de mogelijkheid om een *server push* [ServerPush] te implementeren. Zie volgend puntje voor meer uitleg hieromtrent, aangezien we het er nog vaak over gaan hebben.

HTML Server Push

Hier volgt de ruwe werking van een server push [ServerPush] De server stuurt een stuk data door waarna de browser deze data ontvangt maar hij laat de connectie open; en wanneer er nieuwe data gestuurd wordt toont de clientbrowser dit, zonder telkens de connectie af te sluiten. De server kan deze connectie afsluiten wanneer hij weet dat alle data is gestuurd of de client

kan dit doen wanneer hij de connectie onderbreekt. Het grote voordeel van de server push is zijn efficiëntie en waargenomen snelheid. De client moet niet steeds *pollen* naar nieuwere versies van de data, wat een deel van de middelen van de clientcomputer gebuikt. Wat de snelheid betreft, een push 'duwt' de data niet sneller door het medium maar de connecties moeten niet steeds opnieuw gemaakt worden. De server moet enkel voldoende TCP/IP poorten hebben.
Grootste nadeel is ongetwijfeld dat enkel de Netscape browsers vanaf versie 1.1 de server push ondersteunen.

HTML Forms

HTML Forms zijn een eenvoudige manier om interactiviteit te verzorgen tussen gebruiker en Webapplicaties. Het is een tekstveld waar data kan ingegeven worden die dan door CGI-scripts of server programma's wordt verwerkt.

Frames

Een browser venster kan opgebouwd zijn uit deelvensters, zodat er mogelijkheid is lay-out en inhoud in documenten te voorzien. Tussen deze frames en vensters kan er interactie zijn en één HTML applicatie kan meerdere browservensters gebruiken.

Image Maps

Verschillende delen van een beeld of foto worden aan verschillende URLs verbonden, zodat de gebruiker bij het aanklikken van een deel van dat beeld naar een andere locatie wordt gezonden. Er zijn twee soorten *image maps*, namelijk server- en clientzijde: bij *server-side*, dat ondersteund wordt door de meeste browsers, worden de coördinaten naar de server gestuurd, terwijl bij de *client-side* de map in het HTML-document wordt verwerkt. Dit is enkel in nieuwere browsers geïntegreerd en geeft een feedback aan de gebruikers over welke URL zal gevolgd worden.

Helper applications

Deze applicaties die meestal onder vorm voorkomen van *plug-ins*, verwerken, tonen en spelen documenten af die de browser niet aankan. Het MIME type zal ervoor zorgen dat de juiste applicatie gebruikt wordt, zoals Microsoft Word of Acrobat Reader. *ActiveX controls* zijn ook een voorbeeld van zo'n additionele technologie.

Er zijn drie soorten plug-ins, namelijk de *embedded,* die een deel van een document aankunnen, de *full-page* plug-ins en de verborgen (bijvoorbeeld voor geluid).

DHTML

Dynamic HTML of DHTML, een term om aan te duiden dat bepaalde HTML dynamisch is, bestaat niet op zijn eigen. Het is dus eigenlijk een *buzzword*, niet vastgelegd door het WWW consortium, dat DHTML eens omschreef als een term gebruikt door verkopers om een combinatie aan te duiden van HTML, een *scripting* taal en *Style Sheets*, die ervoor zorgt dat webpagina geanimeerd kunnen worden.
In HTML 4.0 werden er twee nieuwe concepten geïntroduceerd die DHTML mogelijk maakten: CSS en DOM. '*Cascading Style Sheets'* of CSS vormen een model voor stijl en lay-out van HTML, terwijl het '*Document Object Model'* (DOM) een model is voor de inhoud van HTML documenten Het geeft toegangsgelegenheid voor elk element in het document, men kan het bekijken als een soort map.
Verder bespreken we het derde element van de DHTML, de scripting taal.

JavaScript

JavaScript [JavaScript] is een scripting taal, dit is een lichtgewicht computer taal, ontwikkeld door Netscape met een syntax die lijkt op die van Java. Ze is niet statisch getypeerd en er is *type checking*, er kunnen expressies, functies en *flow control* stukken code in geschreven worden. Ze wordt door de client geïnterpreteerd met *run-time object binding*. Het is een objectgeoriënteerde taal, evenwel zonder klassen en overerving; met objecten, eigenschappen en methoden. Objecten kunnen gedefinieerd en uitgebreid worden. JavaScript werkt in alle nieuwere bekende browsers (vanaf versie 3.0).

VBScript

VBScript [VBScript] is ook een scripting taal maar ontwikkeld door Microsoft, als tegenhanger van JavaScript (van Netscape) en vereist Internet Explorer! Het is een lichte versie van Visual Basic, een programmeertaal van Microsoft. Ze wordt ook onmiddellijk tussen de HTML geschreven en geïnterpreteerd door de browser, is ook dynamisch getypeerd.
Erg bruikbaar zal deze taal dus niet zijn, vermits ze de voordelen die het web biedt teniet doet door niet platformonafhankelijk te zijn en er vervangers met equivalente kracht aangeboden worden.

Java

Java [Java] is een objectgeoriënteerde taal met klassen, methoden en *inheritance*. Ze wordt aan serverzijde gecompileerd tot 'bytecode' en er is dus een sterke typering met een statische binding. Onder de vorm van 'applets' kunnen stukken code in een HTML document geplaatst worden. Dit zorgt voor een gebruik van pop-up vensters en een complete interactiviteit en lay-out vrijheid in de applet. Er is wel een strengere beveiliging dan bij java, locale schijven, printers, gebruikersgegevens mogen niet aangesproken worden.
Verschillen van Java t.o.v. JavaScript is dat de initiële moeite, klassendefinities, compilatie, enz zwaarder zijn, het blijft beperkt tot zijn eigen ruimte in de HTML pagina maar de byte code is kleiner en sneller te downloaden. Bovendien wordt het sneller uitgevoerd en heeft het een grafische *toolkit*. De widgets bieden ook meer controle dan HTML forms met JavaScript. JavaScript kan wel aan alle publieke variabelen en methoden in Java en omgekeerd. Java kan ook plug-ins bedienen zoals filmpjes afspelen.

Flash

Flash [Flash] is een multimedia grafisch softwarepakket waarmee men, buiten animaties (waarvoor het oorspronkelijk bedoeld was) nu ook interactieve en geanimeerde websites kan creëren. Dankzij de uitgebreide programmeeromgeving kunnen zowel webdesigners als ontwikkelaars erin sterke sites maken. Het gebruikt vector grafieken, wat betekent dat zijn gegevens minder zwaar zijn dan *bitmaps* en zonder kwaliteitsverlies uitgerekt/ingekrompen kunnen worden. Voordelen van Flash ten opzichte van Java applets en geanimeerde beelden om dynamische effecten te creëren zijn dat Flash sneller laadt en interactief is en geen programmeerkennis vereist. Dankzij de prototypegebaseerde objectgeoriënteerde taal *ActionScript*, die gebaseerd is op JavaScript, kunnen programmeurs er toch krachtige programma's in schrijven.

ShockWave

ShockWave [Shockwave] werd ontwikkeld om interactieve en aantrekkelijke applicaties te creëren, voornamelijk CD-ROM interfaces. Tegenwoordig is de lijn tussen Flash en Shockwave vager geworden en kan men er dus ook productdemo's, Internet verkoopsapplicaties, spelletjes, muziek, *multi-user* samenwerking, geavanceerde chatprogramma's enz. mee ontwikkelen. In tegenstelling tot de kleinere Flash broer die op het animeren van het Internet mikt, is Shockwave meer applicatiegericht. ShockWave werkt via een gratis plug-in, de Shockwave Player, die veel zwaarder en wat eenvoudiger te installeren is dan de Flash omgeving.

Curl

Curl [Curl] is een programmeertaal ontworpen voor de ontwikkeling van interactieve Internet applicaties. Het combineert het gemak van *markup* talen met de functionaliteit van objectgeoriënteerde talen. Specifieker streeft Curl naar het nabootsen van de karakteristieken van HTML, JavaScript en Flash (Macromedia) – allemaal binnen één taal. Een belangrijke eigenheid van Curl is dat de broncode wordt gezonden naar de clients, aangezien dit minder tijd inneemt dan doorsturen van gecompileerde code.
Hoewel de taalspecificatie nu nog niet vrijgegeven is, heeft het bedrijf beloofd delen van de technologie *open source* te maken. Ondertussen is de Curl ontwikkelingsomgeving vrij beschikbaar voor persoonlijk gebruik.

ActiveX

ActiveX [ActiveX] is een technologie ontwikkeld door Microsoft [Microsoft] en werd dus specifiek geschreven voor Internet Explorer en is goed te bekijken als een concurrent voor de Java Beans. ActiveX is een verzameling van definities en technologieën die ervoor zorgen dat software componenten in verschillende talen in een netwerkomgeving kunnen samenwerken. Ze worden automatisch gedownload en geïnstalleerd en hierdoor is veiligheid een groot probleempunt. Vaak wordt 'VeriSign' gebruikt, dit is een standaard voor het digitaal signeren.

3.2.3 Programmeren aan server kant

De server start een script op en geeft data aan de client, het script genereert output en de server zendt het naar de client. Programmeren aan de serverkant is dus het ontwikkelen van applicatiecode die behandeld wordt door de server, het bieden van updates aan de gebruiker, het customiseren van de applicatie voor de gebruiker, een deel van de applicatie toegankelijk maken tot de gebruiker of data die zich op de *server* bevindt ophalen of wijzigen. We bespreken achtereenvolgens kort de belangrijkste concepten en technologieën op dit gebied.

CGI

'Common Gateway Interfaces' of CGI's [CGI] zorgen voor het programmeren langs server zijde. Het is een standaard voor het interfacen van externe programma's – CGI scripts – met Web servers. De Web server zal als een *gateway* werken tussen de webbrowser en de externe programma's. Het wordt dus gebruikt om gebruikersaanvragen, samen met

andere informatie uit forms over te brengen naar die CGI programma's en output en resultaten terug naar de browser te sturen.
Ze zijn geschreven in talen zoals 'Perl' en 'C' en informatie wordt eraan geleverd via omgevingsvariabelen. Ze gebruiken standaard output om informatie terug te geven.

Periodic polling

Dit wordt gebruikt om na een gegeven tijd de pagina te vernieuwen. In het *Refresh statement* kan men het aantal seconden voor vernieuwing definiëren.
Een Java applet kan ook gebruikt worden om de pagina opnieuw te laden, op de achtergrond gebeurt er dan een test om te zien of de pagina is aangepast. Dit is onzichtbaar voor de gebruiker want er moet immers een onzichtbare applet geïmplementeerd worden en dit gebeurt alleen na een aanpassing.

ASP

Active Server Pages [ASP] staan bekend als een alternatief voor CGI, dus een hulpmiddel om dynamische en interactieve websites te bouwen. Het draait binnen ISS, '*Internet Information Services'*, dat verkrijgbaar is met Windows. Het is dus een Microsoft technologie. Wel bestaan er hulpmiddelen, zoals 'ChiliASP' [ChiliASP] en 'InstantASP' [InstantASP], die ervoor zorgen dat de ASP technologie zonder Windows kan draaien. ASP bestanden zijn zoals HTML bestanden maar ze worden uitgevoerd op de server om dan gewone HTML naar de browser te sturen.
Met ASP kan men dynamisch eender welke inhoud van een webpagina veranderen, aanpassen of uitbreiden. De voordelen van ASP ten opzichte van CGI en Perl zijn eenvoudigheid en snelheid.

3.3 Karakteristieken die invloed hebben

Na de bespreking van enkele – de belangrijkste – technologieën kunnen we bekijken welke zaken er invloed hebben op ontwikkeling van webapplicaties in het algemeen en groupware in het bijzonder. We gaan dus de verschillen aanhalen die er zijn tussen de webtechnologieën en de klassieke technologieën.

Een eerste aspect is ervaring. Aangezien het Internet relatief nieuw is, is de ervarenheid met de technologieën en het medium in het algemeen nog niet uitgebreid. Ook de ontwikkelingshulpmiddelen zijn nog niet volgroeid, de focus begint nu pas echt te verschuiven naar applicaties. We zien dit door het ontstaan van 'Web Services', waaronder we de '.NET' [NET] strategie van Microsoft [Microsoft] en zijn concurrenten kunnen rekenen. Men gebruikt

het Web echter nog steeds meer voor documenten van voor applicaties, de Web Services staan nog in hun kinderschoenen. Bestaande technologieën zijn inhoudsgericht en worden ook eerder ontwikkeld om te voldoen aan visuele vereisten, veel meer dan in klassieke software, aangezien het *marketing* betreft. Publieke websites moeten vechten voor gebruikers en de beste manier om gebruikers te doen komen en terugkomen is door een aantrekkelijk geheel te creëren.

Een tweede aspect is de onvoorspelbaarheid van het Internet. Men heeft immers geen idee van hoeveel en welke gebruikers er zich op het netwerk bevinden en hoe dat netwerk eruit ziet. De transfersnelheden zijn niet te berekenen en hangen af van aantal gebruikers op de site en op verschillende cruciale plaatsen in het netwerk (bottlenecks) en van de server en client Internetverbindingen.

Een ander belangrijk aspect is het feit dat het Web client/server gebaseerd is. We hebben eerder gezien, in punt 2.4.5, dat het versturen van berichtjes in groupware bijna onmogelijk is als men geen broadcasting heeft. Stel dat één van de clients een aanpassing maakt op zijn gegevens en dat dit aan alle andere (n) participanten moet getoond worden. Dan moet er eerst een berichtje naar de server gezonden worden. De server krijgt dit en, indien een server push ondersteund wordt of kan gesimuleerd worden, stuurt dan n berichtjes naar alle anderen. Als ondertussen één van de anderen dit ook doet, dan moeten we weer n berichtjes sturen en met de snelheid dat dit gebeurt over het Web, is er een risico van snel met een enorme wachtrij berichtjes te zitten, zodat aanpassingen al snel veel later bij andere toekomen. Een oplossing hiervoor is dus de granulariteit van de berichtjes zo groot te maken dat het systeem alles kan 'slikken'. Probleem is dat dit kan betekenen dat we geen synchrone aanpassingen kunnen maken. Dit moeten we dus later verder onderzoeken.

Een laatste belangrijke eigenschap van het web is (het probleem met) veiligheid. Virussen, *worms*, *hackers* en *crackers*, … het WWW zit er vol van, mede omdat dit zo omvangrijk en open is. Redenen genoeg dus voor ontwikkelaars om hun systemen zo veilig mogelijk te maken en dit werd bereikt door het bestandssysteem aan clientzijde bijna niet te laten lezen, beschrijven of aanpassen. Er kunnen wel oplossingen gevonden worden en *cookies* bijvoorbeeld, die vaak worden gebruikt om gegevens van de webpagina bezoeker op te slaan. Uiteindelijk is dit in het domein van groupware geen ramp, de gegevens staan vaak centraal opgeslagen en dit kan zonder problemen op de server.

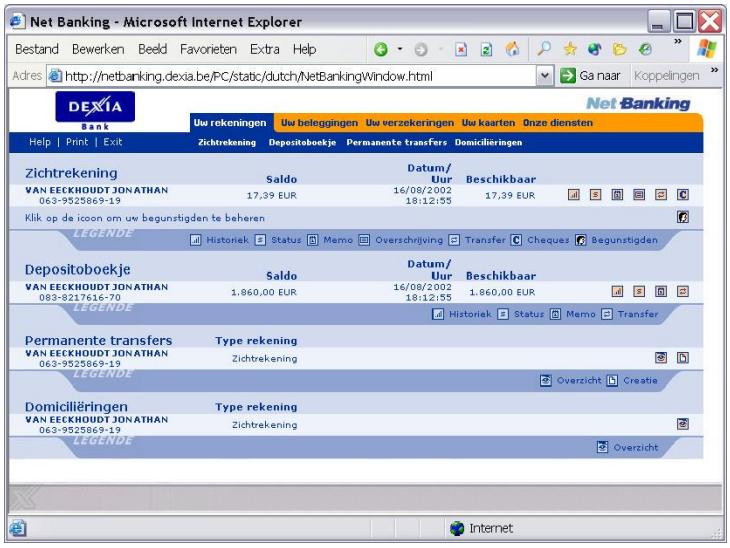
3.3.1 Client consideraties

Hier volgen enkele punten die een belangrijke rol kunnen spelen in het maken van de client kant. Er zijn de problemen met prestaties, compatibiliteit van de browsers en met de technologie en weergave.
Prestaties kunnen afhankelijk zijn van zowel de hardware als de software. Bij de hardware rekenen we de snelheid van het systeem van de gebruiker, geheugen voor cache vooral, schijfsnelheid, netwerk adapter – sommige gebruikers gebruiken immers nog steeds 28.8 modems – en nog meer problematisch zijn monitorresolutie en -kleuren. Dit zorgt voor verschillen in weergave, de schermgrootte en kwaliteit is belangrijk hier. Ook software speelt een belangrijke rol, zoals de te verkrijgen lettertypes, lettertypegroottes en kleuren.
Browsers verschillen ook enorm, niet alleen tussen de platformen en makers maar ook tussen de versies onderling. Het besturingssysteem is ook van belang, gelijke lettertypes op een Macintosh [Macintosh]zijn kleiner dan die op Windows [Windows].

Thin vs. Rich clients

Met de opkomst van websites als interface voor applicaties is er een belangrijke verschuiving aan de gang. Waar men vroeger vaker met *thin clients* werkte, ziet men steeds meet websites die *rich clients* hanteren. In deze paragraaf leg ik kort het verschil en de voor- en nadelen van beide uit.
Thin clients zijn applicaties waar de presentatielaag verdeeld is over server en webpagina. De server creëert een HTML-pagina, met presentatiegegevens erin, die dan doorgestuurd wordt naar de client. De browser vertaalt dan deze pagina en beeldt deze af op het scherm van de gebruiker. Een groot deel van de verwerking van de presentatielaag gebeurt dus op de server, die dus extra wordt belast.
Rich clients daarentegen zijn applicaties waar de presentatielaag zich volledig op de client bevindt. We kunnen hier twee soorten onderscheiden, namelijk de browser gebaseerde en de *standalone* applicaties. Bij de eerste soort speelt de browser nog steeds een belangrijke rol in het weergeven van de gegevens maar vaak zijn er hulpapplicaties die eerst moeten gedownload worden. Dit kunnen bijvoorbeeld applets zijn of een flash movie. De standalone applicaties zijn applicaties die gebruik maken van het Internet om in verbinding te staan met een server. Ze hebben dus geen browser nodig om gebruikt te worden. Een mooi voorbeeld van een rich client die in een browser werkt is ‘Net Banking’ van Dexia [NetBanking], een online bankingapplicatie (zie figuur 3.3-1). Het opstarten duurt iets langer dan een bureaubladapplicatie, mede door de authentificatie, maar eens je binnen bent kan je vlot tussen de verschillende rekeningen, beleggingen, verzekeringen en andere surfen. Het enige wat nog moet gedownload worden zijn de

cijfergegevens, want die kunnen immers ten allen tijde worden aangepast, en nieuwe functies. Hierbij komen we bij de voor- en nadelen van beide soorten clients.



Figuur 3.3-1: Banking applicatie

Wanneer een gebruiker in een thin client een actie onderneemt, bijvoorbeeld op een link klikt, resulteert dit in een relatief lange wachttijd. De nieuwe pagina moet immers volledig worden gedownload. Dit is niet echt een probleem voor websites die louter informatie aanbieden; vaak wordt dezelfde informatie slechts eenmaal bekeken, is het surfgedrag onvoorspelbaar en is het dus niet zinvol om alle gegevens eerst binnen te halen. Rich clients daarentegen kunnen ervoor zorgen dat acties van de gebruiker vrijwel onmiddellijk worden uitgevoerd. We denken hierbij aan het berekenen van een geldbedrag, het controleren van gebruikersinvoer, … . Nadeel is dat alle functies en de verborgen interface eerst moeten worden gedownload maar eens dit in de cache zit gaat het al veel vlotter. Uiteindelijk, als men intensief gebruik maakt van de webpagina, zal er ook minder communicatie met de server aan bod komen dan wanneer men steeds de server raadpleegt om de presentatie te genereren.

3.3.2 Server consideraties

Een belangrijk probleem hier is dat we in een client/server architectuur werken. In de eenvoudige technologieën (dus zonder plug-ins) kan de server kan alleen communiceren met de client als deze er om vraagt (*request)*. Er zijn wel oplossingen hiervoor gevonden en ontworpen. De *client pull* [ClientPull] is er één van, dit is het trucje waar de client na een wachttijd telkens opnieuw de gewenste pagina van de server afhaalt. Deze pagina wordt dat noodzakelijkerwijze vernieuwd. De server push, besproken in 3.2.2, is een aanvulling bij de technologie, spijtig genoeg alleen ondersteund door Netscape.
Ook hier spelen bepaalde aspecten een belangrijke rol ten opzichte van de prestaties. De netwerkconnectie, de hardware en de software spelen hier een cruciale rol. Bij hardware hoort een systeem met snelle schijfresponse en een RAM geheugen dat alle aanvragen aankan. Als er ook programmeren of databankaanvragen zijn moet de processor en het geheugen voldoende zijn. De web server, de databank indien er één is en de software ertussen moeten hier van degelijke kwaliteit zijn.
We gaan ons niet zozeer bezighouden met server aspecten, tenzij het echt gaat om het gedeelte van de collaboratieve applicaties op de server zelf. We zullen ervan uit gaan dat de hardware van hoge kwaliteit is, evenals de software; het probleem dat mensen ondervinden met een server ligt vaker bij de bottleneck in het stuk netwerk juist voor de server dan bij zijn hard- en software.

3.3.3 Netwerk consideraties

Het netwerk is moeilijk of zelfs onmogelijk te controleren. Er zijn twee aspecten waarmee men rekening moet houden en dit zijn de bandbreedte en latentie. Bandbreedte heeft te maken met het maximum aantal data dat tijdens een gegeven tijdseenheid kan doorgestuurd worden. Dit is afhankelijk van het medium waarover het gestuurd wordt en van de hardware die het verstuurt. Telefoondraad heeft een bandbreedte van 64K maar de modem beperkt die tot 56K. De latentie geeft weer hoeveel tijd er nodig is voor een enkel datapakket om de volledige weg tussen twee systemen af te leggen. Dit heeft te maken met netwerkafstand maar ook met de snelheid waarmee informatie door de hardware van de betreffende systemen wordt verwerkt. Latentie kan verkort worden door het gebruik van *mirrorsites* op verschillende plaatsen in de wereld. Let wel dat de netwerkafstand verschillend is van de fysieke afstand, soms kan het zijn dat er omwegen moeten genomen worden wegens defecten of andere blokkades. Netwerkuitrusting kunnen deze latentie ook vergroten.
Tenslotte heeft het gebruik – hiermee bedoel ik de hoeveelheid gegevens die er stromen – van het netwerk ook een invloed op de bandbreedte. Voor het Web is dit moeilijk of zelfs onmogelijk in te schatten: de gegevens worden immers automatisch van router naar router gestuurd, afhankelijk van opstoppingen op stukken van het netwerk.

4 Wat is haalbaar en wat niet?

4.1 Introductie

In hoofdstuk 2 hebben we bekeken wat de eigenschappen en vereisten van raamwerken voor groupware en de verschillende groupware systemen zelf zijn. Vervolgens, in deel 3.2, hebben we de mogelijkheden van de belangrijkste technologieën van het World Wide Web onder de loep genomen. Bedoeling van dit hoofdstuk is om deze twee met elkaar te vergelijken om zo tot de conclusie te komen wat mogelijk is op het Web met groupware en wat niet.

Een eerste opmerking die we kunnen maken is dat de problemen die het Web kent voornamelijk gebaseerd zijn op de prestaties. Gegevens worden verstuurd zonder zekerheid te hebben dat ze aankomen, het Web is onvoorspelbaar wat betreft snelheid en bandbreedte maar bovendien is het model van het Web niet ideaal. In de secties 2.4.5 en 2.4.5 onder 'Scaling up' zagen we dat broadcasting of de ringstructuur een oplossing waren voor het zich opstapelen van berichtjes en dit is niet mogelijk in een client/server structuur zoals die van het Web. Dit zorgt er ook voor dat we de gegevens moeilijk verspreid kunnen bewaren. Cookies zijn een manier om gegevens op te slaan maar deze kunnen (bijvoorbeeld uit plaatsgebrek op de harde schijf van een participant) ten allen tijde door de gebruiker verwijderd worden.
Een laatste punt is de technologie. Waar we in gewone applicaties gebruik kunnen maken van meerdere vensters, menu's en andere widgets moeten we hier, indien mogelijk, andere oplossingen zoeken. Oplossingen zijn pop-up vensters die kunnen zorgen voor het gevoel van het werken systemen met meerdere vensters.

4.2 Mogelijkheden bij groupware systemen

Per type opgesomd in sectie 2.2.3 bekijken we in de volgende paragrafen de mogelijkheden die er zijn om bepaalde systemen te ontwikkelen in een webomgeving en we halen, indien die er zijn, voorbeelden van de reeds bestaande systemen aan.

4.2.1 Computer-mediated communication

E-mail en bulletin boards

Deze soorten applicaties worden ongehinderd op het Web gebruikt aangezien ze asynchroon zijn. Het probleem bij e-mail is de beveiliging; berichten mogen enkel te lezen zijn door de ontvanger(s) voor wie ze bestemd zijn en dit kan men implementeren door o.a. het gebruik van een *login*. Bij forums is de beveiliging een niet zo'n grote zorg maar de identificatie van de gebruikers moet gebeuren zodat iedereen weet welke berichten door welke personen zijn ingezonden. Ook hier is een login de oplossing. Er zijn genoeg voorbeelden van dergelijke systemen, zoals 'Yahoo Mail' [YahooMail] en 'Hotmail' [Hotmail] als voorbeelden van e-mail systemen en forums om op een website te plaatsen zijn vrij te verkrijgen, bijvoorbeeld bij 'Snitz' [Snitz]. Hier hoeven dus niet verder op in te gaan.

Synchrone tekstgebaseerde communicatie

Hoewel de snelheid van gegevensoverdracht hoog genoeg moet zijn, zijn er toch heel wat succesvolle webpagina's die zo'n systeem aanbieden, vooral in de vorm van chatprogramma's. De tekst wordt hier per regel doorgestuurd en bij iedereen getoond op een gezamenlijk transcript. Voorbeelden zijn de site van 'Top Radio' [TopRadio] en die van 'Chat.be' [Chat].
Maar wat als we, zoals het in puntje 2.3.1 eerder besproken 'phone' systeem, karakter per karakter willen doorsturen? Is de snelheid groot genoeg om dit te bereiken? Individuele gesprekken, die normaal in een nieuw venster worden gevoerd, kunnen gemakkelijk in een pop-up worden gevoerd. Dit vormt dus geen probleem en is reeds in de praktijk gebruikt onder andere in de eerder vermelde systemen.

Structured message systems

Ook dit systeem van communiceren gebeurt asynchroon, wat wil zeggen dat het geen probleem is om dit op het Internet te laten gebeuren, men kan dit vergelijken met forums waar de berichtjes gericht zijn naar één of meerdere expliciet gekozen personen. Deze personen zouden hun berichten dan kunnen lezen in hun unieke postvak. We kunnen de structuur van de applicatie bekijken als een combinatie van e-mail en forums.

Video conferencing en communicatie

Cisco [Cisco] biedt een oplossing aan bedrijven om aan IP Videoconferencing te doen, waar data-, stem- en video-verkeer geconvergeerd worden over een gemeenschappelijk pad, vaak een privé-netwerk door Cisco zelf aangelegd. Dit is echter niet vergelijkbaar met systemen die over het WWW zouden moeten werken. Van alle bestaande en wijd verspreide systemen die we op het WWW terugvinden, leunen de zogenaamde 'Webcams' het dichtst bij videocommunicatie. De bedoeling is dat we ergens, op eender welke (publieke) plek op de wereld, een beeld krijgen van een camera die er geplaatst staat, over het Web, meestal het Internet. Een mooi voorbeeld hiervan is de Axis Communications Server Push Page [Axis]. Deze beelden kunt u enkel met de Netscape Navigator [Navigator] bekijken. Er zijn echter reeds een paar verder gevorderde systemen in gebruik, al zijn ze ook nog in hun beginperiode. Informele videocommunicatie, waarbij er frequente korte communicatie is zonder formeel een communicatie te openen of sluiten is bijvoorbeeld onderzocht in de universiteit van Zuid-Parijs door Nicolas Roussel. Hij ontwierp en implementeerde *VideoServer* [Rous99].

We zullen *VideoServer* wat nader bekijken om zo de mogelijkheden ervan met betrekking tot het Web te onderzoeken. Nicolas Roussel baseerde zich op de aspecten die Fish et Al. [Fish et Al.92] ingevoerd hebben en die wij ook aanhaalden in puntje 2.3.1: toegankelijkheid, privacy eenzaamheid.Daarom is het Web ook zo interessant: het is globaal bereikbaar, platformonafhankelijk en het wordt steeds meer een centraal toegangspunt voor applicaties en services. Om *VideoServer* bereikbaar te maken was er de vereiste om geen plug-in te moeten installeren. Hiervoor werd het server push mechanisme, dat sinds 1995 door Netscape in HTML werd voorzien, gebruikt. *VideoServer* is een custom HTTP-server geworden die live of vooraf opgenomen video als JPEG beelden codeert en ze via de server push verzendt. *VideoServer* werd uitgebreid vanuit een CGI-script en is nu een *standalone* server. Het goede eraan is dat de *frame rate* afhangt van de bandbreedte die beschikbaar is. Het telefoon paradigma werd achterwege gelaten. Er is dus geen expliciet

'opnemen van de telefoon' tussen de twee gebruikers; dit zou bisdirectionele audio- en videostromen impliceren.
Problemen met dit systeem zijn onder andere compatibiliteit (de server push werkt immers niet met Internet Explorer [IE]) en de bestaande server en client software, die eerder geoptimaliseerd is voor statische documenten: het verminderen van geheugengebruik en het wijzigen van het formaat van tekeningen kan beter in zelfgemaakte lichtgewicht clients.

Virtuele collaboratieve omgevingen

Dit soort van applicatie is een stuk moeilijker om voor het Web te implementeren. 3D-beelden op het Web brengen is reeds verre van triviaal, dat bovendien combineren met het tonen van alle bewegingen en handelingen van alle gebruikers aan alle andere maakt het alleen maar moeilijker. Een oplossing is om de door te sturen informatie te beperken, immers, als er twee coördinaten doorgestuurd worden zal de client kunnen berekenen hoe de embodiments van de ene naar de andere moeten bewegen.
Het is reeds in de praktijk gebruikt, dit samenwerken in virtuele omgevingen, al is het wel in de nieuwere 3D spelletjes over het Internet ('Quake' en konsoorten).
Technologieën om dit te implementeren kunnen we vinden bij 'Macromedia' [Macromedia'], die 3D én *multi-user* bibliotheken in hun 'Shockwave' hebben voorzien. Voorbeelden hiervan zijn vele spelletjes op www.shockwave.com/games. Het Amerikaanse leger gebruikt ook virtuele werelden om zijn soldaten in te laten trainen. Meer uitgebreide informatie hierover is te vinden in [Sing et Al.99].

4.2.2 Meeting en decision support systems

Argumentatie tools

De synchrone remote versie van de argumentatie tools vergt wat meer moeite om te implementeren dan de asynchrone co-located versie maar aangezien de gebruikers mekaars aanpassingen niet onmiddellijk moeten zien, mogen de blokkeringmechanismen eenvoudig gehouden worden. Een waarschuwingsbericht dat zegt dat een andere gebruiker reeds een aanpassing aanbrengt wanneer een andere dit ook wil doen is voldoende.

Elektronische vergadersystemen

Het is minder logisch om elektronische vergadersystemen onder de vorm van een webapplicatie te creëren, aangezien de gebruikers toch samen zitten. Een mogelijke reden is dat we de applicatie zo overal (zolang we natuurlijk

beschikken over een zaal met erin enkele op het WWW aangesloten Pc's) bij de hand zouden hebben.
Wat moet er doorgestuurd worden in zo'n vergaderruimtes? We hebben gezien dat er een elektronisch whiteboard is, dus moet er voor gezorgd worden dat alle schermen gelijke informatie blijven tonen. Met HTML zal dit niet lukken aangezien er te vaak zal moeten vernieuwd worden, voorbeelden hiervan op het Internet tonen reeds aan dat dit niet evident is, zelfs al is het maar om een één foto per 2 seconden te tonen. We denken hierbij aan online diavoorstellingen of web-cams. Sociale protocollen zorgen ervoor dat de locking weggelaten kan worden, aangezien de spreker vaak controle zal willen hebben over de aanwijzer. Of het vernieuwen van de groep aanwijzer en zijn blokkering snel genoeg zal gebeuren is een vraag die nog open blijft maar het valt te betwijfelen.

Shared work surfaces

Zoals gezien onder sectie 2.3.2 in 'Shared work surfaces' is dit een toepassing van de elektronische vergadersystemen op verschillende plaatsen. Audio en indien mogelijk ook video zouden moeten bijgevoegd worden en bovendien zouden er concurrency control mechanismen aanwezig moeten zijn; de sociale omgang en sociale protocollen (zoals besproken in 2.3.2) zijn er immers niet. Dit alles betekent dat we een grote hoeveelheid data in een korte tijd moeten heen en weer sturen. In 4.2.1 onder het puntje 'Video conferencing en communicatie' hebben we reeds gezien dat er geen echte video mogelijk is over het WWW, enkel een opvolging van beelden, dus dit moeten we al laten wegvallen. Audio zou moeten mogelijk zijn, indien men bijvoorbeeld gebruik maakt van de straffe compressietechnieken die men gebruikt voor de menselijk stem. Deze zijn dankzij het ontstaan van de mobiele telefonie ver gevorderd. GSM's sturen de spraak door aan een snelheid van 13 kbits/sec, wat een zeer klein deel is van de bandbreedte die breedband aanbiedt (+-512kbits/sec). Natuurlijk kunnen we die snelheid slechts uitzonderlijk in zijn geheel bereiken maar spraak zou mogelijk moeten zijn. Aangezien een eenvoudig elektronisch vergadersysteem reeds moeilijk te gebruiken valt over het internet, zullen we aannemen dat de shared work surfaces niet haalbaar zijn op het WWW.

4.2.3 Gedeelde applicaties en artefacten

Gedeelde Pc's en gedeelde venster systemen

De bedoeling van dit type applicatie is om aan een bepaalde gebruiker een zicht te geven een applicatie of zelfs op het volledige systeem van een andere gebruiker. Dit kan gepaard gaan met een gedeelde of volledige controle erover. In een applicatievenster moet hier dus die applicatie of dit systeem nagebootst worden. Met moderne technologieën zou dit geen probleem mogen zijn en aangezien zo'n systemen reeds bestaan op het Internet (bijvoorbeeld 'Netmeeting'[Netmeeting]) is gebleken dat dit medium voldoende zou moeten zijn om deze data over te sturen. Natuurlijk is werken met *sockets* niet hetzelfde als werken in de WWW shell. Audio en vooral video, die best meegestuurd worden om de interactie en het onderling begrip tussen participanten te bevorderen, zullen hoogstwaarschijnlijk teveel van het netwerk eisen.
De gebruiker die een beeld op het werkblad van de andere gebruiker heeft, moet zo snel mogelijk feedback van zijn handelingen zien maar de feedback kan pas teruggegeven worden als de andere computer reageert op die handelingen. Omdat we de applicatie niet kunnen en hoeven na te bootsen, is de eenvoudigste manier om dit te doen gewoon *screenshots*, foto's van het scherm of het deel van het scherm door te sturen. We kunnen dit vergelijken met *streaming* video, alleen de veranderingen van beeld tot beeld worden doorgestuurd. Eigen ervaring heeft geleerd dat de overdrachtsnelheid hoog moet zijn om duidelijke en grote beelden door te sturen. Als we een applicatie *peer-to-peer* delen zitten we al op het randje van onbruikbaar, over het WWW zal dit niet te gebruiken zijn.

Gedeelde verwerker

Het probleem hier ligt hem in het feit dat de granulariteit van de aanpassingen zo klein mogelijk moet zijn, als men samen een tekst verbetert zouden die aanpassingen vrijwel onmiddellijk moeten doorgegeven worden aan elkaar, vooral als de gebruikers op hetzelfde deel van de tekst werken. Als we ook nog met hetzelfde zicht willen werken zouden de schuifbalkgegevens ook moeten doorgegeven worden, met een blokkeringmechanisme voor de schuifbalk. Een gedeelde aanwijzer en audio- en videokanalen waren ook opties voor deze applicatie.
Als we bijvoorbeeld een aanpassing willen doen die slaat op de hele tekst, zoals een bepaald woord vervangen of een opmaak aanpassen, is het misschien niet zo handig om de hele pagina te laten vernieuwen. DHTML of nog beter Java applets bieden hiervoor dan ook een goede oplossing, frames

of pop-up vensters kunnen een alternatief bieden voor de menu's en knoppen die bij standaard tekstverwerkers aanwezig zijn.

Co-authoring systemen

Aangezien de verschillende gebruikers hier op verschillende plaatsen in de tekst werken (de basistekst was een Hypertext structuur) kunnen we een ruwere vorm van blokkeren hanteren. Het feit dat we ook geen WYSIWIS hanteren zorgt ervoor dat dit eenvoudiger te implementeren is. Voor de systemen waar er een meer nauwkeurige samenwerking is kunnen we dezelfde opmerkingen toepassen als bij de gedeelde verwerkers.

Gedeelde agenda's

Dit zal geen probleem zijn om in de WWW Shell te implemeneten, het is immers ook een asynchroon systeem. Het valt te vergelijken met een forum, onderwerpen zijn dan afspraken.
De privacy kan met eenvoudige login worden ingebouwd, zoals we reeds gezien hebben bij de e-mail systemen.
De algoritmen die gebruikt worden staan aan serverkant, dus die vormen ook geen probleem.

4.3 Mogelijkheden bij de shared state modellen

Hier zal ik de drie hoofdmodellen voor het bewaren van een gedeelde toestand uit 2.4.3 bekijken in verband met hun haalbaarheid in de WWW shell, zoals ik voordien gedaan heb met de verschillende groupware systemen.

Centrale opslagplaats

De centrale opslagplaats is de meeste gebruikte vorm van het bewaren van informatie op het WWW, dit zal dus geen significante problemen stellen om te implementeren. Hieruit kunnen we ook afleiden dat het WWW het meest geschikt is om informatie op een (tamelijk) consistente manier te bewaren, zonder dat er al te veel aan moet veranderd worden. Eén groot verschil met de centrale opslagplaats is er wel en dit is dat, hoewel we ook met cache werken, er geen verwittiging naar de participanten wordt gestuurd wanneer deze cache wordt aangepast. Dit zou immers een veel te groot dataverkeer eisen. Daarom dat we er ook niet van uit kunnen gaan dat we met 100% consistente informatie werken. Wanneer we het principe van de centrale opslagplaats correct willen toepassen zullen we de cache moeten uitschakelen ofwel server push moeten invoeren.

Frequente toestandsgeneratie

Om de frequente toestandsgeneratie te implementeren stuiten we al op een eerste groot probleem. De standaard WWW technologieën ondersteunen nauwelijks een server push, laat staan een broadcast. Het beste dat we kunnen doen is een simulatie hiervan door de participanten te laten pollen naar aanpassingen. Omdat er zovele zijn zal dit systeem een grote bandbreedte vereisen. We kunnen dus aannemen dat dit niet echt geschikt is voor in een webomgeving.

Dead reckoning

Dead reckoning is een systeem dat niet veel bandbreedte vereist en in dat opzicht uiterst geschikt is om over een WAN te worden gebruikt. Wat de implementatie op het WWW ervan betreft, is er een probleem, namelijk het feit dat elke deelnemer zijn wijzigingen in status doorstuurt naar alle anderen, wat hier dus niet rechtsreeks kan, de deelnemende pc's zijn immers niet onderling verbonden. Hoogstens kan er een simulatie (die duidelijk niet even krachtig zal zijn) ontworpen worden waarbij elke participant zijn toestand naar de centrale server stuurt, op een plaats waar de anderen regelmatig eens naar aanpassingen komen kijken. Hoe vaak 'regelmatig' nu juist inhoudt moet worden bekeken naargelang het type systeem en bovenal het type netwerk.

4.4 Besluit

De grootste problemen die we ondervinden wanneer we met het Web werken slaan nog steeds op de prestaties die het biedt, hoewel breedband nu aan democratische prijzen wordt aangeboden. Algemeen bekeken kunnen we alle asynchrone applicaties zonder al te veel problemen creëren, het zijn de synchrone die we verder moeten bekijken.
De technologieën die we moeten gebruiken vormen ook weer een kleine hindernis, de basistaal HTML kan daar waar nodig aangevuld worden met andere (dynamische) technologieën die even goed door de recentere webbrowsers worden ondersteund.

Hieronder in Tabel 1 volgt een samenvatting van sectie 4.2, de mogelijkheden die er waren om groupware systemen te implementeren in de WWW shell.

Tabel 1

Type Groupware systeem	**Waarom/waarom niet haalbaar op WWW?**
E-mail en bulletin boards	Asynchroon, vele voorbeelden te vinden en wijd verspreid op het WWW.
Synchrone tekstgebaseerde communicatie	Vele sites voorzien een chatfunctie, is gemakkelijk te ontwikkelen en uitvoerig gebruikt over het WWW.
Structured message systems	Door zijn asynchrone aard heeft het geen grote bandbreedte nodig en bovendien eenvoudig te implementeren in een Webomgeving.
Video conferencing	Te zwaar om over het WWW te implementeren, vele voorbeelden van webcams proberen reeds de limiet te bereiken ondanks slechts enkele beelden per minuut.
Argumentation tools	Eenvoudigweg te implementeren, zie onder 4.2.2.
Elektronische vergaderruimtes	Niet echt zinvol dit te implementeren over het WWW, bovendien te veel dataverkeer.
Shared work surfaces	Te zwaar om te laten draaien over het WWW, vooral door doorsturen van tekeningen.
Gedeelde Pc's en gedeelde venstersystemen	Te zwaar om te laten draaien over het WWW door doorsturen en van screenshots.
Gedeelde verwerkers	Moet mogelijk zijn over het WWW, juist granulariteit moet gevonden worden
Co- authoringsystemen	Zelfde opmerkingen zijn van toepassing als bij de gedeelde verwerkers; ruwere vorm van blokkeren
Gedeelde agenda's	Geen significante problemen, aangezien ze asynchroon werken.

5 Ontwerp van een web gebaseerd systeem

5.1 Inleiding

Nadat we de verschillende grote types collaboratieve systemen hebben besproken in hoofdstuk 2, het Internet in hoofdstuk 3 en de haalbaarheid in hoofdstuk 4, gaan we in dit hoofdstuk een ontwerp van een systeem bekijken.

Wanneer we bestaande programma's bekijken of als we een zoekopdracht ernaar ingeven in een zoekmachine, zien we dat er niet veel gedeelde tekstverwerkers bestaan. Daarom heb ik besloten een ontwerp te maken van een gedeelde tekstverwerker op het web, met technologieën die bruikbaar zijn in (recente en bekende) Internetbrowsers. Hierbij ontwijk ik de zware plug-ins en probeer alleen met DHTML te werken.

5.2 Definitie van een tekstverwerker

Hoewel we al kort gekeken hebben wat een gedeelde tekstverwerker is en welke problemen ermee gemoeid zijn in sectie 2.3.3, gaan we nu een volledigere definitie geven. Zo kunnen we de vereiste functionaliteit en gebruikersinterface opstellen en zodoende verder in dit hoofdstuk een 'verlanglijstje' opstellen.

Een **tekstverwerker** is een computerprogramma dat een gebruiker toelaat om tekst (karakters en nummers, beide gecodeerd door de computer en zijn in- en uitvoer instrument. De karakters zijn zo gearrangeerd om een betekenis te hebben voor andere gebruikers of computers) in te voegen, aan te passen, op te slaan en vaak ook af te drukken. Gewoonlijk geeft een tekstverwerker een "leeg" scherm weer (of een verschuifbare venster) met een vaste regellengte en zichtbare regelnummers. Vervolgens kan men de regels invullen met tekst, regel per regel. Een speciale commandoregel zorgt ervoor dat de gebruiker kan springen naar een volgende pagina, naar voor of naar achter schuiven, globale aanpassingen maken in het document, het document opslaan en andere acties ondernemen. Nadat een document opgeslagen is kan men het afdrukken of weergeven. Voor men dit doet kan men het formatteren voor een specifiek uitvoerapparaat of een type van uitvoerapparaten. Tekstverwerkers kunnen gebruikt worden om broncode van een programmeertaal of documenten als technische handleidingen te creëren.

Een **gedeelde tekstverwerker** is een tekstverwerker die zich bewust is van het feit dat hij collaboratief, door verschillende gebruikers, wordt gebruikt. Hij kan dus verschillende insertiepunten voorzien en biedt blokkeermechanismen die afgestemd zijn op het gedrag van de verwerker.

5.3 Aspecten van gedeelde verwerkers.

In deze sectie bespreken we – wederom uitgebreider dan in sectie 2.3.3 – de voornaamste kenmerken van een gedeelde verwerker waarmee moet rekening gehouden worden. Op elke van deze problemen zullen we dan in de volgende sectie een antwoord geven, in functie van het WWW.

Een belangrijk hulpmiddel in de tekstverwerker is de **commandoregel**. Hiermee kunnen we acties op het volledige document ondernemen, zoals afdrukken en formatteren van de tekst. Hieronder een overzicht van commando's die volgens mij zouden moeten aanwezig zijn in een tekstverwerker.

- Nieuw (leeg) document maken
- Openen – sluiten van een document
- Opslaan document – opslaan als document
- Afdrukken van het document
- Knippen – plakken – kopiëren van delen van het document
- Zoeken – volgende zoeken in het document
- Letteropmaak (lettertype, vet, cursief, onderlijnd, kleur, grootte)
- Alineaopmaak (links uitlijnen, centreren, rechts uitlijnen, inspringen links, inspringen rechts)
- Opsommingtekens

'Ongedaan maken' laten we voorlopig uit deze implementatie wegvallen, dit is niet onontbeerlijk en vereist toch complexe structuren (*history lists* aan clientzijde) en er is vaak reeds een eenvoudige aanwezig in de browser.
De **concurrency control**. Wanneer verschillende gebruikers op dezelfde gegevens werken, moet er voor gezorgd worden dat hun werk niet interfereert. Vooral wanneer men aan een zelfde stuk tekst werkt via netwerken waar er zich significante vertragingen voordoen kan dit irriterend zijn. Gebruiker A zal dan woorden intypen en enige tijd later woorden van gebruiker B ertussen zien verschijnen, zodat de tekst helemaal betekenisloos wordt. Als het netwerk snel genoeg is zullen de gebruikers snel inzien dat er iemand anders op een bepaalde regel bezig is of zullen ze zelfs samen, om beurt, stukken zin kunnen neerschrijven. Maar zelfs in dat geval moeten er bepaalde problemen worden vermeden. Als iemand bijvoorbeeld de letter 'a' in kolom 31 typt en iemand anders (voor hem) in dezelfde kolom de letter 'b', dan zou de 'b' respectievelijk 'a' bij de andere in kolom 32 komen. Met

andere woorden de volgorde van die letters is juist omgekeerd. Hoe gaan we zo'n tekst dan opslaan zodat er geen ambiguïteit is?
Om de eerder beschreven problemen te vermijden moeten we aangepaste **blokkeringmechanismen** invoeren. Deze mechanismen zorgen er dus voor dat slechts één gebruiker aan de volledige tekst of een stuk ervan kan werken, dit noemt men locking. In een gedeelde applicatie kunnen we twee grote soorten locks onderscheiden, de **automatische** en de **expliciete locks**.
Het verschil is eenvoudig. Expliciete locks moeten ingeschakeld worden wanneer een gebruiker een gedeelte van de tekst wil aanpassen, automatische locks worden automatisch ingeschakeld wanneer een gebruiker een aanpassing uitvoert. Wanneer we met automatische locks werken, moeten we ook beslissen wanneer de *lock timeout* zich voordoet. Dit is de tijd die er moet verstrijken voordat zo'n gedeeld voorwerp wordt vrijgelaten. Let wel, hoe sneller dit gebeurt hoe hoger de doorstroomsnelheid van het netwerk moet zijn. Een gedeelde muiswijzer die reeds na enkele seconden wordt vrijgelaten bijvoorbeeld, moet zeer snel nadat een gebruiker hem aansprak feedback geven. Stel dat het netwerk enkele seconden vertraging heeft, dan zal de participant die de wijzer wou gebruiken zien dat deze niet reageert op zijn commando's, zodat hij het misschien opgeeft omdat hij denkt dat de wijzer in gebruik is terwijl hij de controle wel had.
Expliciete locks zijn handig wanneer een gebruiker niet het risico wil lopen om tijdens een korte denkpauze zijn lock kwijt te zijn, zodat hij onverwacht niet kan verder werken aan zijn stuk tekst omdat iemand anders erin beginnen te werken is. Expliciete locks zijn ook voordeliger bij een traag netwerk, omdat de gebruiker eerst zal wachten tot hij feedback krijgt over de aanvraag van zijn lock, zonder dat hij al nutteloos zitten werken heeft.

Wanneer we met verschillende mensen op een zelfde gedeeld gegeven werken, moeten we er voor zorgen dat het zicht dat verschillende gebruikers hebben op de tekst onderling consistent blijft. Dit heeft ook te maken met het blokkeren van gedeeltes van tekst. We moeten voorkomen dat twee gebruikers op dezelfde plaats schrijven en zo verkeerde data te zien krijgen, wat vooral gevaarlijk is als we verschillende exemplaren van het tekstbestand bewaren. Wanneer we het verschuiven van tekst delen kunnen zich andere problemen voordoen. Als we bijvoorbeeld toelaten om twee gebruikers te laten werken op een stuk tekst dat zich op hetzelfde deel van het scherm bevindt, is het niet ondenkbaar dat ze al snel zoveel tekst intypen dat dit niet meer op hetzelfde scherm kan getoond worden. Dan zou het heel lastig zijn als de gebruiker die het recht over de schuifbalk niet verworven heeft moet wachten met zijn stuk tekst tot de andere gedaan heeft en bereid is de schuifbalk af te staan.

Een interessant gegeven waarmee we moeten rekening houden is de actie 'ongedaan maken'. Wanneer we dit willen invoeren moeten er aan de clientzijde **history lists** voor zo'n acties aanwezig zijn. Bijkomend probleem is natuurlijk dat zo'n *undo* gevolgen kan hebben indien de gebruiker hiermee de aanpassingen van iemand anders zou wijzigen.

Laatste punt is de **communicatie** tussen de gebruikers onderling en de **awareness** of het zich bewust zijn van andere gebruikers en hun aanpassingen. Hoe en hoe snel stromen aanpassingen van de andere participanten naar elkaar door? Wanneer een gebruiker een globale actie moet uitvoeren op het document, kan en zal dit gevolgen hebben op de anderen. Een eenvoudige manier om te verwittigen maar ook om tot een overeenkomst te komen over bijvoorbeeld hoe een paragraaf te herfraseren, moet dan voorzien zijn, een beter communicatiemiddel dan bijvoorbeeld de telefoon. We kunnen moeilijk iedereen één voor één opbellen om zo tot een besluit te komen over de globale opmaak. Met betrekking tot deze vereiste, stelt zich de vraag of er gesprekken kunnen gevoerd worden tussen een deel van de gebruikers of worden alle boodschappen naar iedereen gestuurd?
Een goede gedeelde tekstverwerker moet naar mijn mening meer bevatten als enkel een tekstverwerker die door meerdere mensen tegelijk kan worden gebruikt. Het moet ook een toepassing bevatten waarmee mensen kunnen communiceren en om de bandbreedte zoveel mogelijk te sparen, kunnen we hier een zeer eenvoudige chat- of discussieruimte aan toevoegen. Zoals de meeste bekende chatruimtes moet elke participant naar één, naar meerdere of naar alle andere gebruikers kunnen schrijven. Elke gebruiker moet ook kunnen zien welke andere gebruikers er zich aanmelden om er zo ermee te communiceren maar nog meer, elke gebruiker moet kunnen zien wie een lock waar heeft aangebracht. Zo kunnen gebruikers zie wie welke aanpassingen doet en afspraken maken met de juiste persoon.

5.4 Doelstellingen van de tekstverwerker op het web

Eigenschappen van een tekstverwerker: De applicatie moet minstens de belangrijkste eigenschappen bezitten om ze te kunnen beschouwen als een tekstverwerker. Hierbij denken we onder andere aan de acties die in 5.3 bij de commandoregel werden opgesomd.

Collaboratie van verscheidene gebruikers: De tekstverwerker moet gebruikers toelaten om samen bestanden te verwerken zonder fysische nabijheid. Alle gebruikers moeten een consistent zicht hebben en de aanpassingen zo snel mogelijk zien gebeuren; gebruikers moeten ten allen tijde in of uit de groep kunnen gaan.

Vertrouwde handelingen: Hoewel de verwerker niet zal ontworpen worden als een bureaubladapplicatie, moeten de commando's indien mogelijk intuïtief zijn, gebruikers zouden niet te veel tijd nodig mogen hebben om ermee te leren werken. Hiervoor zal er een user interface (UI) moeten geïmplementeerd worden die de UI van een traditionele applicatie nabootst.

Redelijke prestaties: Communicatieprotocollen die gebruikt worden moeten ervoor zorgen dat de participanten een consistent beeld hebben van de gegevens, dus dat de aanpassingen zo snel mogelijk worden doorgestuurd. In eerste plaats moet een gebruiker zijn eigen aanpassingen zonder vertragingen zien gebeuren, zodat hij kan werken zonder storing.

Fout tolerantie: De groepssessie moet, ondanks machinecrashes of mensen die de groep aansluiten of verlaten, vlot draaien.

5.5 Ontwerp van de tekstverwerker

Nadat alle eisen en doelstellingen werden uitgeschreven, ga ik nu over naar het ontwerp (ontwerpdocument: zie Appendix) van de gedeelde tekstverwerker. Hierin zal ik zowel de user interface en de functionaliteit van de applicatie bespreken, als enkele specifieke oplossingen voor de problemen die hogerop genoemd zijn. Deze versie is losjes gebaseerd op het principe van de gestructureerde coöperatieve tekstverwerker Alliance [Deco et Al.96]. Alliance vereist dat elke gebruiker een webserver heeft draaien, zodat op die manier van beide kanten aanvragen tot aanpassen kunnen geïnitieerd worden. Deze tekstverwerker gebruikt het server push mechanisme om de

modificaties of nieuwe alinea's door te zenden. Bovendien is er een manier tot communicatie voorzien.

Aangezien we op het WWW met een client/server structuur werken, zullen we verplicht zijn een centrale kopie van het document te bewaren. De participanten kunnen dan een kopie afhalen, terwijl hun veranderingen zo snel mogelijk worden doorgegeven. Het grootste probleem hier is dat we ervoor moeten zorgen dat de aanpassingen kunnen worden doorgegeven (feed through) binnen een redelijk tijdsbestek en zonder een andere gebruiker zijn werk te verstoren. De snelheid waarmee dit gebeurt zal eigen zijn aan de internetsnelheid.

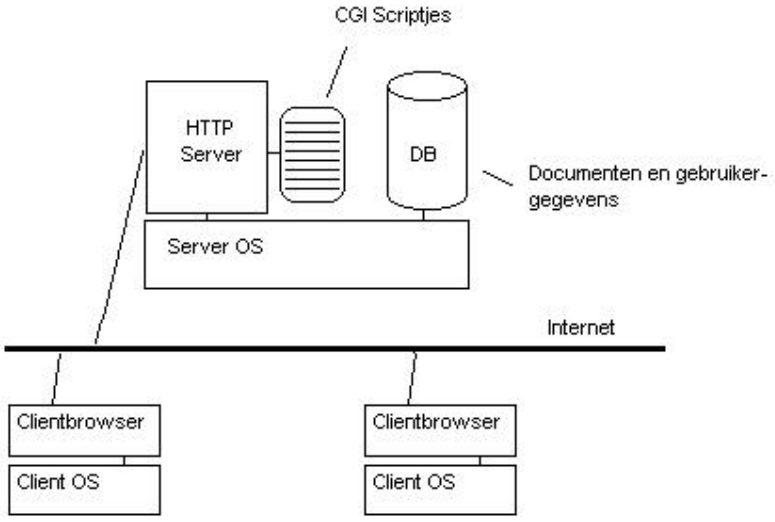
5.5.1 Technologieën

We hebben reeds vaak het server push mechanisme aangehaald. Hier leggen we dit mechanisme verder uit.. Op de server moet een CGI scriptje uitgevoerd worden, dat aangeroepen wordt door de client. Wanneer een client een aanvraag doet om een server push connectie te openen, met andere woorden het CGI script aanspreekt, stuurt dit script berichtjes van het type "multipart/x-mixed-replace", een variant van het "multipart/mixed" bericht. Deze "multipart/mixed" is een MIME type dat wordt gebruikt door een HTTP-server om antwoorden te sturen op een aanvraag van een client.
Hier is een voorbeeld van zo'n "multipart/x-mixed-replace" bericht, met wat uitleg erbij.

```
Content-type: multipart/x-mixed-replace;boundary=EenString
-- EenString
Content-type: text/plain
Data voor het eerste object.
-- EenString
Content-type: text/plain
Data voor het tweede en laatste object.
-- EenString --
```

Hierboven hebben we twee keer een dataobject, met als type 'text/plain' (dit moet elke keer in de *header* worden gestopt en het type van de data kan dus telkens verschillen) naar de client gestuurd. Wat zo schitterend is aan dit type berichten, is dat ze nooit hoeven te eindigen. De client kan wel stoppen met ontvangen wanneer hij maar wilt.
Buiten deze server push maken we ook gebruik van de HTTP 'Post' methode, waar de client via een URL en parameters een aanvraag doet naar een script op de server. Verder in deze sectie wordt duidelijk wanneer we welke connectie gebruiken.
Met JavaScript kunnen we deze dataobjecten opvangen, analyseren en verwerken. Onze applicatie heeft twee types objecten nodig, namelijk

aanpassingen op het document en tekstberichten voor in het communicatiedeel. Hiervoor kunnen we elk dataobject beginnen met een type-header, die zo kort mogelijk moet zijn om zoveel mogelijk bandbreedte vrij te houden voor gegevens.
Onze pagina bestaat uit enkele frames, waarin zich verschillende pagina's bevinden die met elkaar kunnen communiceren via de onderliggende JavaScript. Ze kunnen apart ververst worden.
Twee van die frames worden continu aangepast met behulp van de JavaScript. De inkomende data wordt ermee opgevangen, verwerkt en in het juiste frame geplaatst.
Documenten bestaan uit 'rich text', dit is tekst met opmaak in HTML. Zo al het eenvoudig zijn om tekst aan te passen, weer te geven (HTML wordt omgezet in opmaak) en op te slaan in de databank.



Figuur 5.5-1 Architectuur van de verwerker

De architectuur van dit model is weergegeven in figuur 5.5-1. Het systeem werkt in een browser die JavaScript ondersteund, dus volgens het client/server model. De server moet CGI-scripts kunnen aanbieden, een databank hebben (en die natuurlijk ook kunnen aanspreken) en een minimum van connecties (ongeveer 50-tal) aankunnen zonder te grote netwerkvertragingen te ondervinden.

5.5.2 Werking van gehele programma

Aanmelden

Om de verschillende documenten te beschermen, moet elke gebruiker zich aanmelden. Bij de administrator kan hiervoor een login een paswoord worden verkregen, waaraan één of meerdere tekstbestanden zijn verbonden. Deze login heeft nog een ander doel. Het identificeert namelijk elke gebruiker en zo kunnen we controleren en andere gebruikers meedelen wie wat aanpast heeft en wie met wie communiceert. In Tabel 2 hebben we zo'n voorbeeld van een gebruiker.

Tabel 2

Login	jveeckho@vub.ac.be
Paswoord	*********
Bijnaam	Jona
Informatie*	Jonathan Van Eeckhoudt; 2de lic info VUB

Een login is eenvoudigweg te implementeren in HTML-code of via een CGI programma. Elke gebruiker die met de applicatie wilt werken, krijgt dus van de beheerder zo'n uniek login, met een bijhorend paswoord. Als login kunnen we best een e-mail adres gebruiken. Dit is automatisch uniek en zorgt ervoor dat elke gebruiker automatisch bereikbaar is buiten de applicatie (tenzij hij natuurlijk een vals adres opgeeft). Het paswoord is kan verschillend van het paswoord van zijn e-mail account zijn, dit hangt van de gebruikers zelf af. De bijnaam, dit is de naam die aan anderen getoond wordt, kan de gebruiker zelf bepalen. Deze hoeft ook niet uniek te zijn maar indien twee gebruikers met dezelfde bijnaam op hetzelfde document werken, kan dit voor verwarring zorgen. De gebruikers zien dit probleem zelf wel, dus moet de server hiermee geen rekening houden. Deze gegevens worden opgeslagen in een databank op de Server. Wanneer een gebruiker een foutieve login en/of paswoord ingeeft, kan hij niet verder gaan en mag (moet) hij opnieuw proberen.

Opvragen gewenste document

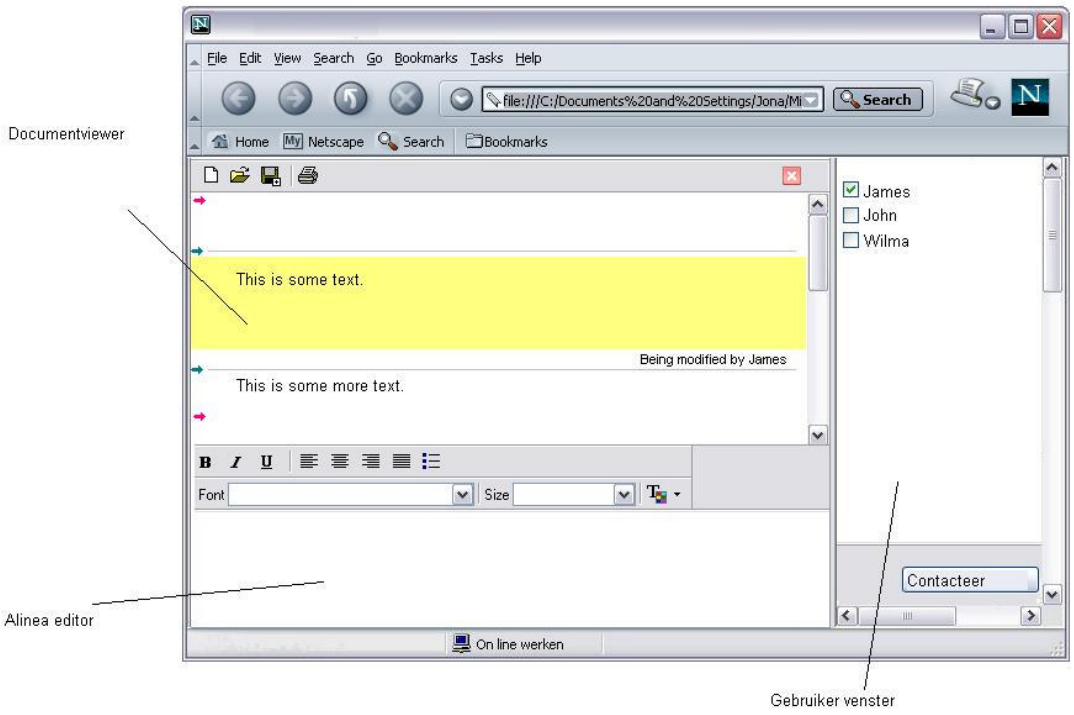
Na het aanmelden krijgt de gebruiker een eenvoudige lijst van links naar bestaande documenten. Hier kan hij ook een leeg document starten of zijn gegevens bekijken/wijzigen. Wanneer hij op één van de links klikt wordt er (via een Post) een document getoond, wordt er eerst gevraagd naar een titel die kan gegeven worden aan een leeg document of komt hij op de pagina van gebruikergegevens terecht.

Gebruikergegevens pagina

Op de gegevenspagina kan de gebruiker zijn persoonlijke gegevens bekijken en/of aanpassen. Er is een 'OK'-knop en een 'annuleren'-knop. Dit is een pagina met tekstvelden zoals men ze op de meeste websites met aanpasbare gebruikergegevens vindt .

Verwerkerpagina

Hier krijgen we in eerste instantie twee frames te zien(zie figuur 5.5-2). Rechts is er een frame met alle aanwezige gebruikers. Dit frame wordt d.m.v. de server push continu aangepast. Wanneer we een gesprek willen starten met één of meerdere gebruikers, moeten we de bijhorende *checkbox* aankruisen en op de knop 'Start gesprek' klikken. Dan krijgen alle participanten een dialoogvenster waar we kunnen converseren.



Figuur 5.5-2 Verwerker venster

Aan de linkerkant hebben het document venster, dat via de server push en de JavaScript continu wordt aangevuld. We kunnen maar één document per keer openen. We zien de paragrafen als *tekstvelden* die niet door de gebruiker zelf aanpasbaar zijn. Bij die tekstvelden is er steeds een link naar de gebruiker die de bijhorende alinea aanpast en dus de lock heeft geplaatst of een keuze om zelf de alinea te blokkeren. Tussen twee alinea's kan er ook een nieuwe geplaatst worden, door op de tekening te klikken.
Hier is reeds een deel van de commandoregel te zien, met acties die niet echt aanpassingen vereisen op het document zelf. Een overzicht:

- **Nieuw (leeg) document**
  Zorgt voor de aanmaak van een document op de server, met de titel die meegegeven wordt door de maker. Deze persoon verdwijnt van het scherm bij de andere participanten en krijgt een nieuw (leeg) document te zien.

- **Openen – sluiten**
  Zorgt dat de gebruiker die deze actie onderneemt een bepaald document krijgt of terug naar het document keuzevenster gaat. Hij verdwijnt van het scherm bij andere gebruikers.

- **Opslaan – opslaan als**
  Opslaan hoeft niet voor het hele document, wel voor een alinea. Dit wordt bereikt door de knop 'Invoegen' in te drukken. 'Opslaan als' maakt een kopie op de server van een bepaald document aan, waar deze gebruiker een versie van te zien krijgt.

- **Afdrukken**
  Aangezien elk document uit stukken bestaat en op het venster verschillende frames worden getoond, moeten we naar een printvriendelijke weergave gaan om te kunnen afdrukken. De client maakt een collage van alle stukjes tekst op een welbepaald moment en toont dit in een venster.

- **Knippen – plakken – kopiëren**.
  Plakken is geen probleem, zolang we het doen van een (al dan niet geblokkeerde) alinea naar en een geblokkeerde alinea. Ook geïntegreerd in de browser.

- **Zoeken – volgende zoeken**
  Is wederom reeds geïntegreerd in de browser.

Wanneer we een alinea willen creëren klikken we op de juiste tekening en wordt er een nieuwe alinea aangemaakt op de server. Onze alinea krijgt dan een uniek ID, een nummer dat gegeven wordt door de server. Deze nummers zorgen ervoor dat de het document kan worden samengesteld in de vorm van een boom. Als we bijvoorbeeld alinea 1.1 en alinea 1.2 hebben en een gebruiker wil ertussen een nieuw creëren, dan wordt dit 1.1.1. Dit is belangrijk wanneer we de stukken doorsturen, elk stukje tekst moet tussen de andere stukjes kunnen geplaatst worden door de clients.
Wanneer een gebruiker een alinea wilt bewerken, sturen we een aanvraag tot blokkeren op naar de server, samen met het alinea-ID. Indien we eerst zijn wordt dit doorgezonden naar alle participanten, ook naar de initiërende verwerker. Deze krijgt dan een nieuw frame in beeld, nadat de exacte inhoud van de alinea is ontvangen via een antwoord op een *request*. Het kon immers zijn dat de server geen tijd had om alle aanpassingen door te sturen. In dit frame bevindt zich ook een commandoregel waarmee paragraafgebonden acties kunnen ondernomen worden. De letteropmaak (lettertype, vet, cursief, onderlijnd, kleur, grootte), de alineaopmaak (links uitlijnen, centreren, rechts uitlijnen, inspringen links, inspringen rechts) en de opsommingtekens zijn in

deze commandoregel verwerkt. Met JavaScript kunnen we de gegevens dynamisch aanpassen, we werken immers met HTML tekst.

5.5.3 Serverzijde

De server moet inkomende aanvragen aankunnen, zowel met de Post als met de server push methode. De server zal ook twee wachtrijen moeten hebben; één waar alle aanpassingen wachten op het verzenden en één waar de communicatie wacht. Dit om een verschil te kunnen maken in prioriteit tussen de communicatie en de continue aanpassingen naar de client toe. Als een client een alinea wil aanpassen, doet hij een post naar de server om zo rechtstreeks de laatste versie van de alinea te krijgen. Het is immers mogelijk dat de aanpassingen niet allemaal doorgestroomd zijn, bijvoorbeeld bij dataverlies of overbezetting van het netwerk. De client bewaart de status, dit is mogelijk omdat we niet zoveel clients toelaten (op een coöperatieve verwerker zitten immers doorgaans niet meer dan tien mensen gezamenlijk te werken). Hij houdt dus bij wie er een conversatie heeft met wie, wie er momenteel een alinea bewerkt en wie er aanwezig is. In geval van crash van de client is het nodig dat er een timeout is, zodat hij er niet van uit gaat dat ze nog aanwezig zijn.
Zie ook appendix

Conclusie

Wanneer we het uiteindelijke resultaat van de verwerker bekijken is het een variant op de chatapplicatie geworden; een applicatie die zijn strepen reeds heeft gehaald op het internet. Het is een eenvoudige maar krachtige applicatie, waar een beperking aantal gebruikers de reden is tot succes. Erg is dit niet, aangezien we toch niet met een twintigtal personen een document kunnen verwerker De communicatie tussen server en clients zal zeker niet die van een grote chatapplicatie overschrijden, dus het dataverkeer moet haalbaar zijn in een omgeving met breedband. Er werden toch een paar voorzorgsmaatregelen genomen om de gevolgen van vertragingen en pakketverlies te minimaliseren. Inconsistenties tussen de verschillende clients zijn bijvoorbeeld toegelaten. Elke gebruiker krijgt zijn gewenste feedback door het schrijven in een apart frame, zonder gestoord te worden door trage connecties.
In deze applicatie hebben we geprobeerd een rich client te creëren met eenvoudige webtechnologieën, zonder te berusten op Java-applets, Shockwave of andere zware plug-ins. Voordeel is bijvoorbeeld dat we geen vensters moeten nabootsen en downloaden, zodat we de gebruiker minutenlange wachttijden van staren op een grijze rechthoek aanbieden. DHTML en vooral server push, een technologie die volgens mij ten onrechte in de vergeethoek is geraakt, kunnen met wat inventiviteit, een lichtgewicht vervanging vormen. Grootste nadeel van deze applicatie is ongetwijfeld het niet compatibel zijn met andere webbrowsers dan Netscape, terwijl platformonafhankelijkheid toch een vereiste was. Gelukkig is Netscape wel beschikbaar voor alle besturingssystemen en bovendien gratis te installeren.
Het ontwerp van de gestructureerde coöperatieve verwerker is bovendien ook toe te passen op bovenvermelde talen.

Referenties

Boeken

[Cone et Al.98] *Coordination Technology for Collaborative Applications: organisations, processes, and agents*
door Wolfram Conen en Gustaf Neumann (eds.)
1998 Springer
ISBN 3-540-64170-X

[Dix et Al.98] *Human-Computer Interaction*
door Alan J. Dix, Janet E. Finlay, Gregory D. Abowd en Russel Beale
2de editie; 1998 Prentice Hall
ISBN 0-13-239864-8

[Gamm et Al.95] *Design Patterns:Elements of Reusable Object-Oriented Software*
door Erich Gamma, Richard Helm, Ralph Johnson en John Vlissides
1995 Addison-Wesley
ISBN 0-20-163361-2

[Sing et Al.99] *Networked Virtual Environments: Design and Implementation*
door Sandeep Singhal en Micheal Zyda
1ste editie; 1999 Addison-Wesley
ISBN 0-20-132557-8

Artikels

[Bouv00] *Designing User Interfaces for Collaborative Web-Based Open Hypermedia*
door Niels Olof Bouvin
2000 Proceedings of the ACM 2000 Conference, pp 230-231

[Chan et Al.95] *On Computer Supported Collaborative Writing Tools for Distributed Environments*
door Kai H. Chang, Yu Gong, Tim Dollar, Shefali Gajiwala, Byong Lee en A. Wesley Wear
1995 Auburn University

[Stef et Al.87] WYSIWIS Revisited: early Experiences with Multiuser Interfaces
door M. Stefik, D.G. Bobrow, G. Foster, S. Lanning en D. Tatar
1987 ACM Transactions on Office Information Systems, pp 147-167
http://www2.parc.com/istl/members/stefik/colab.htm

[Deco et Al.91] *Structured Cooperative Authoring on the World Wide Web*
door Dominique Decouchant, Vincent Quint en Manuel Romero Salcedo
1991
http://www.w3.org/Conferences/WWW4/Papers/91/

[Deco et Al.96] *Alliance: A Structured Cooperative Editor on the Web*
door Dominique Decouchant en Manuel Romero Salcedo
1996 ERCIM workshop on CSCW and the Web
http://orgwis.gmd.de/projects/W4G/proceedings/alliance.html

[DTro00] *Cursus 'Ontwerpmethoden voor Internetgebaseerde systemen Hoofdstukken 1,2 en 3*
door Prof. Dr. Olga De Troyer
2000 VUB

[Dix96] *Challenges and Perspectives for Cooperative Work on the Web*
door Alan Dix
1996 ERCIM workshop on CSCW and the Web
http://orgwis.gmd.de/projects/W4G/proceedings/challenges.html

[Feis00] *Posties : A Webdav Application for Collaborative Work*
door Joachim Feise
2000 University of California
http://pv182069.reshsg.uci.edu:8080/~jfeise/UCI/Research/Posties/Posties.pdf

[Fiel00] *Architectural Styles and the Design of Network-based Software Architectures*
door Roy Thomas Fielding
2000 University of California

[Fish en Al.92] *Evaluating Video as a Technology for Informal Communication*
door R. Fish, R. Kraut, R. Root en R. Rice
1992 Proceedings of ACM CHI '92, pp 37-48

[Fu99] *Class Experiences in Writing Collaborative Client/Server Applications for the Internet (Draft in Progress)*
door Zhi Fu, Debarata Ghosh, Hemang Lavana, Lorie I. Moffitt, Steve Nelson, J. Marschall Smith, Jun Zhou en Franc Brglez
1999 CBL in het North Carolina State University

[Girg et Al.96] *Experiences in Developing Collaborative Applications Using the World wide Web "Shell"*
door Andreas Girgensohn, Alison Lee en Kevin Schlueter
1996 ACM Press

[Holt96] *The Futplex System*
door Koen Holtman
1996 ERCIM workshop on CSCW and the Web
http://orgwis.gmd.de/projects/W4G/proceedings/futplex.html

[Knis93] *Issues in the Design of a Toolkit for Supporting Multiple Group Editors*
door Michael Knister en Atul Prakash
1993 University of Michigan
http://citeseer.nj.nec.com/knister93issues.html

[Lee et Al.96] *Developing Collaborative Applications Using the World Wide Web 'Shell' (Tutorial Session)*
door Allison Lee en Andreas Girgensohn
1996 ACM Press

[Prak et Al.92] *Design considerations in choosing operations for building groupware systems*
door Atul Prakash en Michael J. Knister
1992 University of Michigan

[Rous99] *Beyond Webcams and Videoconferencing: Informal Video Communication on the Web*
door Nicolas Roussel
1999 Active Web 1999
http://www.visualize.uk.com/conf/activeweb/proceed/pap15/

[Sefr et Al.] *A Model of Collaborative Writing*
door Jan Sefranek en Milos Kravcik
Comenius University, Bratislava, Slovakia

[VdB et Al.02] *Stageverslag van Webtechnologiën*
door Gregory Vandenbroucke, David Van Damme en Jonathan Van Eeckhoudt
2002 VUB

[VDam02] *Onderzoek naar het ontwikkelen van grafische Gebruikersinterfaces met behulp van moderne Webtechnologieën*
door David Van Damme
2002 VUB

Applicaties en technologieën

[ActiveX]	Hoofdpagina van de ActiveX technologie http://www.microsoft.com/com/tech/activex.asp
[ASP]	Active server pages, technologie van Microsoft http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/iisref/aspguide.htm
[Axis]	Axis Communications Server Push Webcam, werkt enkel in Netscape Navgator… http://libcam.bucknell.edu/cgi-bin/push.html
[CGI]	Pagina met enorm veel uitleg over de CGI http://hoohoo.ncsa.uiuc.edu/cgi/
[Chat]	Belgische 'ontmoetingsplaats' op het WWW, gespecialiseerd in chatten. http://chat.be/
[ChiliASP]	Technologie van Chili!Soft. Is nu Sun One Active Server Pages genoemd. http://www.chilisoft.com/
[Cisco]	Bedrijf gespecialiseerd in netwerk-oplossingen http://www.cisco.com/
[ClientPull]	Technologie van Netscape, beschikbaar sinds Netscape Navigator 1.1 http://wp.netscape.com/assist/net_sites/pushpull.html
[Colab]	WYSIWIS Revisited: early Experiences with Multiuser Interfaces door M. Stefik, D.G. Bobrow, G. Foster, S. Lanning en D. Tatar 1987 ACM Transactions on Office Information Systems, pp 147-167 http://www2.parc.com/istl/members/stefik/colab.htm
[Curl]	Softwarebedrijf Curl http://www.curl.com/
[EdgeOfCreation]	Online Multiple User Domain-systeem http://www.edgeofcreation.net
[Flash]	Technologie van Macromedia, ideaal om rijke websites te creëren http://www.macromedia.com/software/flash/

[Hotmail] Gratis e-mail service op het Web van Microsoft.
http://www.hotmail.com/

[HTTP] HyperText Transfer Protocol; protocol gebruikt om hypertext te versturen.
http://www.w3.org/Protocols/

[HTML] HyperText Markup Language, dé taal om hypertext te publiceren op het Internet.
http://www.w3.org/MarkUp/

[Hypertext] Ontworpen door Ted Nelson in *Literary Machines.* Pagina met uitgebreide uitleg.
http://www.iath.virginia.edu/elab/hfl0037.html

[IBIS] Issues as Elements of Information Systems door W. Kunz en H.W.J. Rittel
1970 Universität Stuttgart

[ICQ] Gratis communicatiesoftware als bureaubladapplicatie.
http://web.icq.com/

[IE]. Webbrowser van Microsoft.
http://www.microsoft.com/windows/ie/default.asp

[InfLens] Intelligent Information Sharing Systems door T.W. Malone, K.R. Grant, F.A. Turbak, S.A. Brobst en M.D. Cohen
1987 Communications of the ACM

[InstantASP] Website van Stryon, ontwikkelaar van Instant ASP
http://www.stryon.com/products.asp?s=1

[Java] Website van de bekende taal van Sun
http://java.sun.com/

[JavaScript] Pagina met tutorials en codevoorbeelden in JavaScript
http://javascript.com/

[Macintosh] Hard- en software bedrijf, gespecialiseerd in volledige computersystemen
http://www.apple.com/

[Macromedia] Softwarebedrijf, bekend om Flash en Shockwave.
http://www.macromedia.com

[Memex] Pagina met het bekende werk 'As we may think' van Vanevar Bush erop, waar de memex uitvoerig wordt in besproken.
http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm

[Messenger] Communnicatiesoftware van Microsoft.
http://msn.messenger.be

[Microsoft] Bekende en groot softwarebedrijf
http://www.microsoft.com

[Mosaic] De eerste webbrowser, gecreëerd in het National Center for Supercomputing Applications.
http://archive.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html

[Navigator] Webbrowser van Netscape.
http://browsers.netscape.com/browsers/main.tmpl

[NET] Hoofdpagina van Microsoft's implementatie van de Web Services: .Net
http://www.microsoft.com/net/

[NetBanking] Online banking applicatie van Dexia
http://netbanking.dexia.be/

[Netmeeting] Communnicatiesoftware van Microsoft.
http://www.microsoft.com/windows/netmeeting/

[Netscape] Softwarebedrijf, gespecialiseerd in Internetpaketten
http://www.netscape.com/

[OE] Outlook Express, e-mail applicatie van Microsoft.
http://www.microsoft.com/windows/oe/

[Quilt] Collaborative Document Production Using Quilt
door M.P.D Leland, R.S. Fish en R.E. Kraut
1988 CSCW '88 Proceedings, pp 206-215

[Rose] Hoofdpagina van de bekende ontwerp software Rose
http://www.rational.com/rose

[ServerPush] Technologie van Netscape, beschikbaar sinds Netscape Navigator 1.1
http://wp.netscape.com/assist/net_sites/pushpull.html

[Shockwave]	Technologie van Macromedia, ideaal om rijke websites te creëren, broertje van Flash http://www.macromedia.com/software/director/
[Snitz]	Softwarebedrijf gespecialiseerd in webforums. http://www.snitz.net/
[Talk]	Standaard commando in een Unix sessie. Het commando verscheen in 4.2BSD.
[TopRadio]	Hoofdpagina van de bekende radiozender, Flash vereist. http://www.topradio.be/
[Vax]	Marquette's VAX is een cluster van vijf Digital Equipment Corporation computers. Vax Phone is een middel dat je toelaat om een interactief real-time gesprek te hebben.
[VBScript]	De gebruikersgids voor VBScript van Microsoft. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vbstutor.asp
[Windows]	Besturingssysteem van Microsoft http://www.microsoft.com/windows/default.mspx
[WindowsXP]	Besturingsysteem van Microsoft. http://www.microsoft.com/windowsxp/default.asp
[WWW]	Universum van informatie die bereikbaar is via netwerk. Onstaan uit visie van Tim Berners-Lee. http://www.w3.org/WWW/
[YahooMail]	Gratis e-mail service op het Web van Yahoo. http://mail.yahoo.com/

Appendix