Software document

Instituut voor Perceptie Onderzoek
Postbus 513, 5600 MB Eindhoven

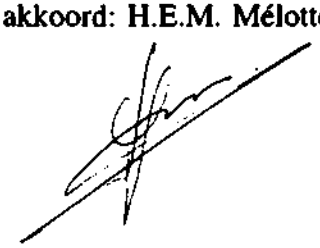
30.01.1996

Rapport no. 1089

Software Document integreren van de grafische user interface met het SoundTablet

D.M.R. Goeloe

Voor akkoord: H.E.M. Mélotte



SOFTWARE DOCUMENT
Integreren van de grafische user interface met het SoundTablet

Auteur : D.M.R. Goeloe

Bedrijfsmentor: Ir. L.H.D. Poll

Schoolmentor : P. van Baalen

Datum : januari 1996

Samenvatting

Dit verslag beschrijft de ontwikkeling van een systeem dat een Grafische User Interface (GUI) integreert met de SoundTablet software. De opdracht is in het kader van EU-project waaraan het Instituut voor Perceptie Onderzoek (IPO) deelneemt waarbij men een naar een mogelijkheid zocht om Grafische user interfaces toegankelijk te maken voor blinde gebruikers.

Voor de ontwikkeling van het gewenst systeem moest er gebruikt worden gemaakt van Object Modeling Techniek (OMT) [Rumbaugh, 1991]. Omdat OMT niet bekend was, moest dit techniek eerst bestudeerd worden, voordat de opdracht uitgevoerd kon worden.

Bij de uitvoering van de opdracht werd er begonnen met een analyse van de situatie. Tijdens deze analyse werd een concreet definitie van het gewenst systeem en de eisen waaraan het moet voldaan opgesteld (user requirements). Hierna werd overgegaan tot het opstellen van een objecten model. In het objecten model zijn alle objecten die bij het systeem relevant zijn, vastgelegd. Het objecten model beschrijft de statische structuur van het systeem. Na het opstellen van het objecten model werd het dynamische model (die de tijdsafhankelijke veranderingen van het systeem illustreert) opgesteld. Het dynamisch model bestaat uit tijds- en toestand-diagrammen (het systeem werd als een eindige automaat beschouwd) . Als laatste werd een functioneel model opgesteld. Dit werd aan de hand van data flow diagrammen vastgelegd. Deze drie modellen geven een compleet beschrijving van het systeem. In hoofdstuk twee wordt in het kort een beschrijving gegeven van het gebruikte methode.

Na het ontwerpen werd overgegaan tot het implementeren van het ontwerp in C++. Vanwege tijd gebrek was het niet mogelijk om het ontwerp volledig te implementeren. Er is veel tijd besteed naar het uitzoeken van informatie m.b.t. Windows applicaties.

Indien aanpassingen aan het systeem aangebracht moeten worden, moet dit document als uitgangspunt dienen voor het aanbrengen van deze aanpassingen. Breng deze aanpassingen eerst in dit document aan alvorens ze te implementeren.

Inhoud

1 Inleiding

Dit verslag is het resultaat van een afstudeeropdracht dat uitgevoerd werd bij het Instituut voor Perceptie Onderzoek (IPO). Dit is een instituut dat zich bezig houdt met onderzoek op het gebied van perceptie. Daar wordt sinds januari 1992 onderzoek gedaan naar het toegankelijk maken van grafische user interfaces (GUI's) zoals Microsoft Windows voor blinde gebruikers. Hiervoor nam IPO van 1992 tot 1993 deel aan een EU-project. Voor dit EU-project is een model ontwikkeld waarmee de inhoud van willekeurige GUI schermen beschreven kunnen worden. Tevens is er gedurende dit project een interface ontwikkeld waarmee het mogelijk is om informatie met de GUI uit te wisselen, welke tijdens de uitvoering fouten maakt en te langzaam is [Poll, 1995].

De user interface waarmee de blinde gebruiker communiceert bestond in het EU-project uit een reeds bestaand systeem ontwikkeld door een van de project partners. Hiermee is het mogelijk om toegang te krijgen tot GUI's. GUI specifieke aspecten zoals directe manipulatie zijn echter niet mogelijk. Daarom heeft het onderzoek zich vanaf 1993 gericht op het toegankelijk maken van deze GUI specifieke eigenschappen waarbij gebruik gemaakt wordt van synthetische spraak, opgenomen spraak, geluid en een absolute muis. Dit onderzoek resulteerde in een systeem waarmee het mogelijk was om de basisprincipes van de ontworpen niet visuele interface te testen. Het systeem, SoundTablet genaamd, is echter gebaseerd op een simulatie van een GUI, omdat er nog geen link is met een bestaand GUI.

Men wenste een systeem dat de reeds in het EU-project ontwikkelde GUI integreert aan de SoundTablet software, inclusief aanpassingen en optimalisatie. Deze aanpassingen en optimalisaties hebben betrekking op zowel de methoden voor ophalen van GUI informatie als het simuleren van muis acties en toetsenbord acties naar de GUI. in dit verslag zie je de ontwikkeling van dit systeem.

2 Software document wegwijzer

Dit verslag vormt het software document dat hoort bij de integratie van de grafische user interface aan het SoundTablet en geeft een beschrijving van de ontwikkeling van de software. Elke fase van ontwikkeling is hierin vastgelegd.

De modelleringstechniek die hierbij wordt toegepast is Object Modeling Techniek (OMT). Dit is een object georiënteerde techniek voor taal-onafhankelijke software ontwikkeling. Deze methode geeft een leidraad voor de requirements, een doeltreffend ontwerp en beter onderhoudbare systemen. Deze techniek is meer dan een manier om te programmeren. Het is een manier om abstract te denken over een probleem door gebruik te maken van concepten uit de werkelijkheid i.p.v. concepten uit de computer wereld. Met OMT wordt een analyse gedaan van een probleem en een oplossing hiervoor ontworpen.

Gebruik makend van OMT heeft de ontwikkeling van het systeem in fasen plaatsgevonden. Eerst vond er een analyse plaats waarin het probleem grondig werd bestudeerd en beschreven (user requirements). Hierna werd in deelfasen het model ontworpen, waarbij elke deelfase een model leverde (m.n. objecten model, dynamisch model en functioneel model) dat een aspect van het systeem beschrijft. Aan de hand van het ontworpen model is verder overgegaan tot implementatie.

2.1 Hoe werk je met OMT ?

Zoals eerder verteld wordt OMT gebruikt om tot een model te komen van het systeem dat men wil ontwikkelen. Door gebruik te maken van OMT word je in zekere zin gedwongen het ontwerpen in fasen uit te voeren. Het resultaat van deze fasen vormt het compleet model van het systeem.

Je begint met een studie van het probleem: aan de hand van de opdracht beschrijving, het ondervragen van de opdrachtgever en het raadplegen van andere documenten waarnaar gerefereerd wordt. In de studie ga je zoeken naar de aanleiding tot de opdracht, je bestudeert de omgeving waarin het systeem zich later gaat bevinden en je onderzoekt aan welke eisen het systeem moet voldoen. Je moet ervoor zorgen dat wat je gaat ontwikkelen, datgene is wat de opdrachtgever wenst. Tevens heb je na deze studie een beter beeld van wat er precies ontwikkeld moet worden en je hebt een verzameling objecten opgesteld die je nodig hebt bij het opstellen van een objecten model.

Uit de verzameling objecten die je hebt opgesteld, kies je die objecten die relevant zijn voor het systeem. De keuze van deze objecten blijkt uit de studie die uitgevoerd is. Met de gekozen objecten wordt het objecten model gebouwd. In het objecten model worden de relaties tussen de gekozen objecten onderling en hun attributen beschreven. Het objecten model illustreert

dus de statische structuur van het te ontwikkelen systeem. Door deze statische structuur goed te bestuderen, krijg je een beter begrip van het systeem. Aspecten die te maken hebben met tijd en veranderingen komen hier niet naar voren. Deze horen bij het dynamisch model.

Behalve de tijd- en veranderingenaspecten worden *gebeurtenissen* (events) die ertoe leiden dat het systeem in bepaalde *toestanden* (status) verkeert ook in een dynamisch model beschreven. Dit is het hart van een dynamisch model, het beschrijven van de reactie van het systeem op bepaalde gebeurtenissen. Door het systeem te beschouwen als een eindige automaat kun je met behulp van toestand diagrammen dit laatste goed illustreren. Alvorens het toestand diagram op te stellen, is het handig om eerst een tijdsdiagram op te stellen waarin je een sequentie van gebeurtenissen aangeeft. Het tijdsdiagram kun je gebruiken om voor elke gebeurtenis een deel van het toestand diagram te tekenen dat een toestandsovergang aanduidt. Kortom het dynamisch model moet beschrijven, **wanneer** er een gebeurtenis optreed.

Wat er gebeurt wordt door het functioneel model beschreven. Het functioneel model beschrijft hoe de output van een systeem berekend wordt uit de input, zonder rekening te houden met de volgorde waarin dit gebeurt. Het functioneel model bestaat uit data flow diagrammen die de invoerstroom, de operaties die op deze invoer worden uitgevoerd, de interne opslag van data en de uitvoerstroom aangeven. Uit het functioneel model leid je de diverse operaties (functies) af die tot de uitvoer leiden.

Het functioneel model is het laatste gedeelte van de modelleringsfase. Na het opstellen van het functioneel model is er een complete beschrijving van het systeem tot stand gekomen. Het functioneel model beschrijft wat er gebeurt (hoofdstuk 6), het dynamisch model beschrijft wanneer het gebeurt (hoofdstuk 5) en het objecten model beschrijft met wie het gebeurt (hoofdstuk 4). In de genoemde hoofdstukken zijn de drie modellen te zien die samen een ontwerp vormen van het systeem. Met dit compleet model kan men overgaan tot het implementeren van het systeem.

Een sterk punt van het gebruik van OMT (of een soortgelijke modelleringstechniek) is, dat bij toekomstig onderhoud van het systeem dit document goed van pas komt indien iemand anders dan de ontwikkelaar dit moet doen. Men gebruikt het document om zich te oriënteren, om te kijken waar de aanpassingen gebracht moeten worden, deze aan te brengen en vervolgens implementeren. Dit geeft tijdwinst en voorkomt onnodige inspanning.

3 User requirements

Hier wordt uitgelegd wat er ontwikkeld moet worden en aan welke eisen voldaan moet worden. Dit document fungeert als startpunt van waaruit de verdere ontwikkeling van het systeem wordt voortgebracht. Indien er tijdens de ontwikkeling nieuwe eisen ontstaan, zal dit document aangepast moeten worden. Hetzelfde geldt voor eisen waaraan het systeem niet meer hoeft te voldoen. Hiertoe wordt er een beeld gegeven van de manier waarop men tot nu toe het omschreven probleem (in de inleiding) heeft opgelost en waarom dit geen ideale oplossing is.

3.1 Huidige situatie

Men is al een tijdje bezig geweest om GUI's toegankelijk te maken voor blinden. De tot nu toe bedachte oplossingen zijn allemaal softwarematig. Sommige van deze oplossingen zijn aparte applicaties voor blinden, bijvoorbeeld een tekst editor voor blinden [Edwards, 1987]. Het probleem hiervan is dat blinden die zich in een werkomgeving bevinden met niet blinden, dezelfde applicaties willen gebruiken die hun niet blinde collega's gebruiken. Een andere oplossing is een zogenaamde "software bridge". Dit type software vangt elke aanroep die een systeem doet van een of andere GUI routine en zet daarbij de executie van zijn eigen I/O routines. Enkele voorbeelden van dit type software zijn, "Outspoken" van Berkeley Systems en "Slimware Window Bridge" van Syntha-voice Computers.

Een groot nadeel van een software bridge is, dat deze alleen voor een bepaalde GUI toegepast kan worden. Wordt er een nieuwe versie van de betreffende GUI gemaakt, dan moet er meestal een bijpassende software bridge ontwikkeld worden. Het gevolg is dat de blinde gebruiker moet wachten op de nieuwe versie software bridge voordat hij/zij met de nieuwe versie GUI kan werken, terwijl zijn/haar niet blinde collega's al werkzaam zijn met de nieuwe versie van de GUI. Verder heb je voor verschillende GUI's verschillende software bridges nodig, waarbij elke software bridge zijn eigen user interface heeft voor blinden. Dus moet de blinde gebruiker telkens met een nieuwe user interface leren werken.

3.2 Systeem definitie

Door nu software en hardware te combineren probeert men de genoemde nadelen uit de weg te ruimen. Tijdens het EU-project waaraan IPO deelnam, heeft men zo'n oplossing bedacht. De hardware waar het hier omgaat wordt een SoundTablet genoemd (zie figuur 3.1). Met deze hardware en de nodige software worden de GUI objecten in de vorm van geluid, opgenomen spraak en synthetische spraak aan de blinde gebruiker gepresenteerd. Het SoundTablet bestaat

uit een vierkante elektronische mat met daarop een muis die zich gedraagt als een scanner. Terwijl je deze beweegt worden er geluiden gegenereerd die de bewegingen uitdrukken. Op deze manier wordt onderscheid gemaakt tussen tekstregels en graphiks. Iconen kunnen worden gerepresenteerd door alledaagse geluiden. Een regel tekst kan gelezen worden door bijvoorbeeld op een knopje te drukken of je kunt een regel tekst laten spellen door de

Figuur 3.1 SoundTablet

muis/scanner op de betreffende regel te bewegen, waardoor elke karakter een voor een wordt gelezen.

Met de muis/scanner kunnen ook iconen geselecteerd of geactiveerd worden en je kunt de cursor plaatsen op de plek waar je een stuk tekst wil invoeren. Zelfs het slepen met objecten is mogelijk. Door bijvoorbeeld het geluid van het schuiven van een zware doos weet de gebruiker dat hij/zij aan het slepen is. Kortom, door constante geluid feedback weet de gebruiker waar hij/zij mee bezig is.

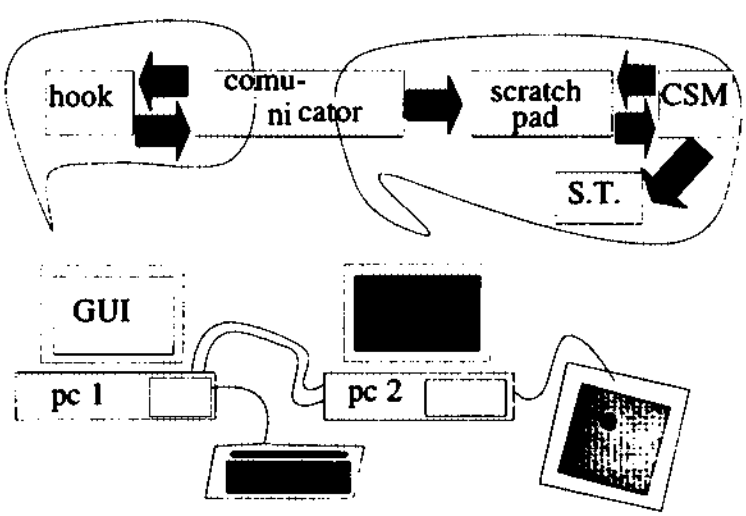
Voor de functionaliteit van het SoundTablet heeft men de nodige software ontwikkeld (of is men nog bezig met ontwikkeling). Er is een software ontwikkeld HOOK genaamd, die functies bevat waarmee je nagaat of er veranderingen hebben plaatsgevonden (nieuwe objecten) op het beeldscherm waar de GUI draait. Deze veranderingen (herkende objecten) dienen op het SoundTablet gebracht te worden zodat de SoundTablet inhoud compatible is met de beeldscherm inhoud. Dit wordt door de Current Screen Module (CSM) [Gerrits, 1993] gedaan. De herkende objecten worden in een zogenaamd Scratch Pad geplaatst, waarna de CSM die objecten gebruikt om de SoundTablet inhoud opnieuw te bouwen. De manier waarop de objecten in de Scratch Pad zijn vastgelegd, is anders dan de manier waarop diezelfde objecten herkend worden. Dus moet het herkend object alvorens het in de Scratch Pad te plaatsen (zodat de CSM de SoundTablet inhoud kan bouwen), in een voor de CSM

begrijpelijke vorm vertaald worden.

Je kunt je voorstellen dat het niet mogelijk is om dit alles door een pc te laten doen. Het eist veel machine kracht. Voor de hierboven beschreven opzet is er een tweede pc nodig waarop de Scratch Pad en de CSM en de SoundTablet software (die het SoundTablet bestuurd, genereren van geluiden e.d.) draaien, terwijl op de andere pc een GUI en de software die objecten herkend draaien. Nu is er een software nodig die de herkende objecten overdraagt van de ene machine naar de andere (via de seriële poort), zodat de CSM z'n werk kan doen. Dit is degene wat hier ontwikkeld wordt. Deze software wordt de Communicator genoemd. Behalve het overdragen van de objecten zorgt de Communicator voor de vertaling van de herkende objecten. Je kunt de Communicator als volgt definiëren (zie figuur 3.2):

Systeem definitie:

Het overdragen van beeldscherm informatie van de ene module naar de andere.



figuur 3.2 Systeem configuratie

3.3 Systeem eisen

Hieronder staan enkele eisen waaraan de Communicator moet voldoen. Deze eisen definiëren de taken van de Communicator.

Definities:

prototype: Het SoundTablet en de diverse softwares (subsystemen) die voor de correcte werking ervan zorgen (de hele figuur 3.2).

systeem: De software die hier ontwikkeld wordt (Communicator) die een onderdeel is van het prototype.

externe systemen: Dit zijn die softwares (subsystemen) die ontwikkeld zijn (of in proces van ontwikkeling zijn) door derden.

Systeem eisen:

1) Het systeem is een onderdeel van een uiteindelijk prototype
Zoals in de systeem definitie is uitgelegd bestaat het prototype uit diverse subsystemen die de werking van het prototype voortzetten. De Communicator is een onderdeel hiervan.

2) Overdracht van informatie door de Communicator vindt plaats wanneer deze informatie gegenereerd is door een extern systeem.
De Communicator is in contact met twee externe systemen. Een van de externe systemen heeft behoefte aan informatie die de andere bezit. Deze informatie dient door het systeem (Communicator) overgedragen te worden.

3) De Communicator heeft de rol van interpreter tussen de externe systemen.
Elk extern systeem legt de nieuwe informatie anders vast. Het systeem moet er daarom voor zorgen dat de gewenste informatie op de juiste wijze wordt gepresenteerd aan het betreffend extern systeem.

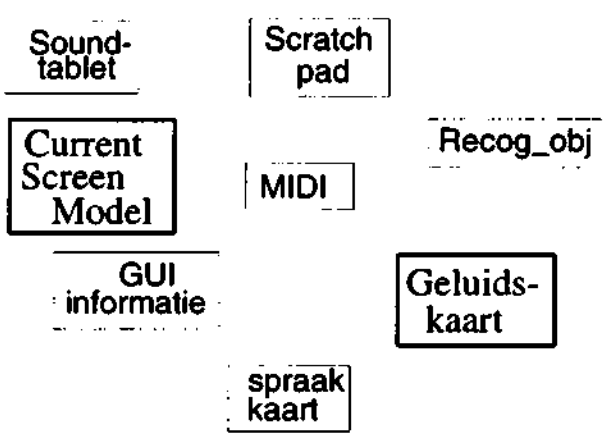
4) De Communicator handhaaft de real time werking van het prototype
Het prototype heeft een real time karakter, dus de systemen moeten ervoor zorgen dat het prototype een real time werking heeft.

4 Objecten Model

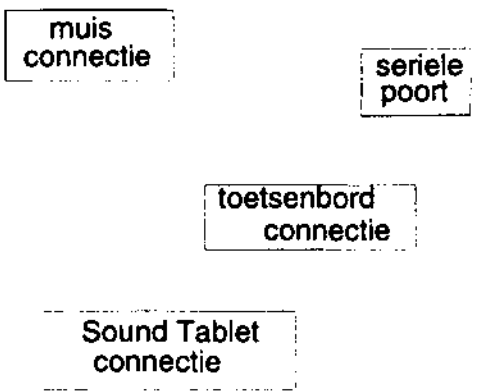
Dit is een beschrijving van de structuur van de objecten van het systeem (Communicator). Het objecten model vormt het skelet waarop het functioneel- en dynamisch model gelegd wordt. Hiervoor moeten de objecten, hun onderlinge relatie en hun attributen, geïdentificeerd worden.

4.1 Hoe komen we tot de objecten ?

Ten eerste moeten de objecten die relevant zijn voor dit systeem geïdentificeerd worden. Door een verzameling van objecten op te stellen kun je nagaan welke objecten bij het te ontwikkelen systeem relevant zijn. Deze verzameling objecten stel je op tijdens de analyse van de opdracht. De elementen van de genoemde verzameling haal je uit de opdracht beschrijving, de diverse documenten die betrokken zijn bij deze opdracht, door de opdrachtgever te ondervragen of uit eigen ervaring bij vroegere opdrachten. In figuur 4.1 en 4.2 zie je de verschillende objecten die gevonden zijn na de bestudering van de opdracht. In figuur 4.1 vind je de objecten die overgenomen zijn uit de diverse documenten en in figuur 4.2 vind je de objecten die zelf bedacht zijn.

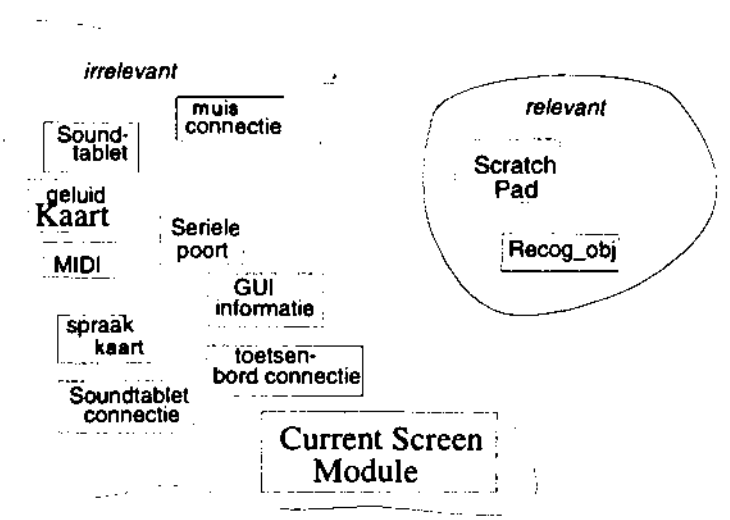


Figuur 4.1 Kandidaten voor object klasse



Figuur 4.2 Afgeleide objecten

Het doel is om uit deze verzameling objecten de goede kandidaten voor object klassen te kiezen. Dit gebeurt door eliminatie van objecten die niet van toepassing zijn voor het systeem. Figuur 4.3 laat zien welke objecten geëlimineerd zijn en welke objecten gekozen zijn voor object klassen.



Figuur 4.3 Splitsing tussen relevante en irrelevante objecten voor het systeem

Na bestudering van de verschillende objecten en de probleem beschrijving, zijn de volgende objecten gekozen voor object klasse:

- Scratch Pad (Scrp_obj)
- Recog_obj.

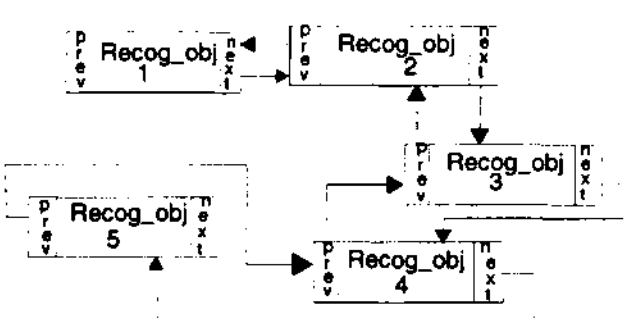
Wat deze objecten zijn en waarom ze gekozen zijn, wordt in de hierna volgende paragrafen duidelijk gemaakt.

4.2 Data dictionary

De namen van de gekozen objecten zeggen niet veel over hun eigenlijke rol die ze binnen het systeem (Communicator) gaan spelen. Daarom wordt er hier voor elk object een beschrijving gegeven.

Recog_obj:
Objecten die ontstaan na transformatie van de objecten die m.b.v. HOOK herkend zijn.

Scratch pad:
Dit object bevat lijsten met kopieën van Recog_obj met daaraan toegevoegd pointers waarvan een naar het volgend element en het ander naar het vorig element van de lijst verwijzen (linked list). Figuur 4.4 geeft een beeld hiervan.



Figuur 4.4 Lijst van herkende objecten (Scratch pad)

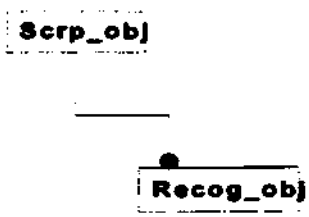
Aan de hand van deze linked list kan de Current Screen Model (CSM) de inhoud van het SoundTablet opbouwen, zodat dit overeenkomt met de werkelijke inhoud op het beeldscherm waar de GUI draait. Aan de attributen kun je het verband tussen deze twee objecten zien.

Object	**Attributen**	
Scrp_obj	OCR object name	
	Upper left corner	
	Bottom right corner	
	Text	
	Text_color	
	Next	;pointer naar volgend Scrp_obj in de lijst
	Previous	;pointer naar vorig Scrp_obj in de lijst
Recog_obj	OCR object name	
	Upper left corner	
	Bottom right corner	
	Text	
	Text color	

Waarom zijn juist deze twee objecten gekozen ? Zoals eerder vermeld gebruikt de CSM de linked list (Scrp_obj) om de inhoud van het SoundTablet op te bouwen. De wijze waarop objecten worden herkend komt niet overeen met de wijze waarop deze objecten in *Scrp_obj* worden vastgelegd. Daarom worden de herkende objecten eerst omgezet in het object *Recog_obj*, zodat zij heel eenvoudig in *Scrp_obj* gekopieerd kunnen worden.

4.3 Objecten relatie

Van de gevonden objecten die relevant zijn voor dit systeem (Scrp_obj en Recog_obj), is het object diagram in figuur 4.5 ontwikkeld. Dit diagram geeft een illustratie van de onderlinge relatie tussen de objecten. De relatie tussen *Scrp_obj* en *Recog_obj* is een 1 op veel (of 1 op n) relatie, dat wil zeggen dat het object *Scrp_obj* bestaat uit meerdere objecten *Recog_obj* (dit wordt aangegeven door een zwart cirkeltje aan het eind van de verbindingslijn bij *Recog_obj* in de tekening). Dit houdt in dat het object *Scrp_obj* ten minste dezelfde attributen bevat als het object *Recog_obj* (zie vorig paragraaf).



Figuur 4.5 Object diagram

Nu de objecten geïdentificeerd zijn, kunnen we overgaan tot het opstellen van het dynamisch en het functioneel model van waaruit de operaties op de gevonden objecten bekend zullen worden. Wat we op dit moment hebben is een statische structuur van het systeem.

5 Dynamisch model

Het dynamisch model beschrijft het tijdsafhankelijke gedrag van het systeem en z'n objecten. Het dynamisch model is belangrijk voor interactieve systemen zoals de Communicator. Hier wordt er gekeken naar events (gebeurtenissen) die plaatsvinden en de wijze waarop het systeem zal reageren. Dit wordt aan de hand van toestand diagrammen geïllustreerd.

5.1 Deelsystemen en scenario's

Omdat het systeem de informatie overdracht tussen softwares die op verschillende pc's draaien regelt, wordt het systeem opgedeeld in twee deelsystemen. We duiden deze deelsystemen aan met Sendcom (die voor de omzetting en versturing van informatie zorgt op pc1) en Receivecom (die voor de ontvangst en doorsturen van deze informatie zorgt op pc2). Door het toestand diagram die bij deze deelsystemen hoort te ontwikkelen, wordt het systeem in z'n geheel beschreven. Voordat deze toestand diagrammen opgesteld worden, wordt er eerst een tijdsdiagram (die de sequentie van gebeurtenissen aangeeft) ontwikkeld, wat als uitgangspunt dient voor het opstellen van de toestandsdiagrammen.

5.1.1 Scenario Sendcom

Dit scenario vindt plaats wanneer er een normale interactie bestaat tussen Sendcom en zijn omgeving. Met normaal wordt bedoeld, zonder te kijken naar bijzondere gevallen, zoals bijvoorbeeld bij het optreden van fouten e.d.

- Sendcom polt m.b.v. de functies uit HOOK voor nieuwe beeldscherm informatie (HOOK bevat functies waarmee nieuwe beeldscherm informatie gedetecteerd en opgehaald wordt).
- Sendcom krijgt data van HOOK en zet deze om in Recog_obj.
- Sendcom maakt Recog_obj gereed voor verzending naar pc2.
- Sendcom is bezig met verzending van bytes via de seriële poort.
- Sendcom is klaar met informatie overdracht en wacht op een bevestiging.
- Sendcom ontvangt bevestiging en gaat over tot polling.

Hier volgt een bijzonder scenario:

- Sendcom polt voor data.
- Sendcom krijgt data van HOOK en zet deze om in Recog_obj.
- Sendcom gaat over tot transmissie.

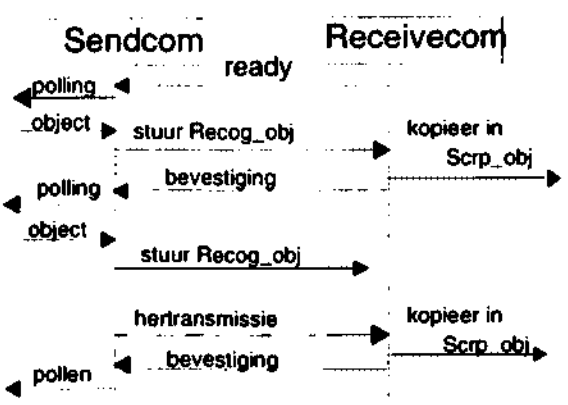
- Sendcom is bezig met data overdracht.
- Sendcom is klaar met data overdracht en wacht op een bevestiging.
- Indien er geen bevestiging is ontvangen, wordt Recog_obj opnieuw verstuurd (TIME OUT) en wacht voor bevestiging.
- Sendcom ontvangt bevestiging en kan dus weer pollen.

5.1.2 Scenario Receivecom

Een scenario voor Receivecom:

- Receivecom wacht op informatie vanuit Sendcom.
- Receivecom krijgt de informatie via de seriële poort binnen.
- Receivecom assembleert het object Recog_obj opnieuw en kopieert het in Scratch Pad.
- Receivecom stuurt een gereed signaal (bevestiging) door naar Sendcom en wacht weer op informatie.

In onderstaande figuur (figuur 5.1) staat een tijdsdiagram dat de beschreven scenario's illustreert.



Figuur 5.1 Tijdsdiagram Sendcom en Receivecom (sequentie van gebeurtenissen)..

De verkorte pijl met een kruisje in de tekening hierboven geeft aan dat er iets mis is gegaan met de transmissie van die informatie, waardoor die niet bij Receivecom is aangekomen. Na een bepaalde tijd wachten op een bevestiging vanuit Receivecom die niet aankomt, wordt deze informatie weer verzonden. Deze wachttijd moet onmerkbaar voor de gebruiker blijven om de real time werking van het SoundTablet te garanderen. Bij een baudrate van bijvoorbeeld 28.8

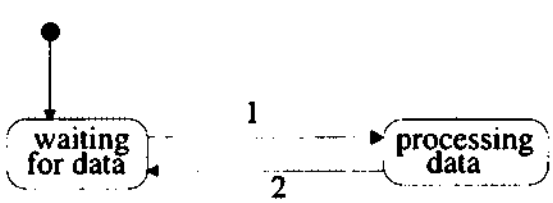
kbaud en waarbij *Recog_obj* een omvang heeft van bijvoorbeeld 50 bytes, kan in 1 seconde zo'n 72 (= 28.8 kbaud/(50*8)) objecten opgestuurd worden door Sendcom naar Receivecom. In dit geval kun je een vrij kort wachttijd kiezen, bijvoorbeeld 3 seconden.

5.2 Toestand diagrammen

Met behulp van de hierboven beschreven scenario's en tijdsdiagram, wordt het toestand diagram voor beide processen (Sendcom en Receivecom) ontwikkeld. Elk pad in het toestand diagram representeert een scenario. Hieronder zie je hoe het toestand diagram verder uitgediept wordt door het toestand diagram voor een bepaalde toestand te genereren. Op deze manier wordt het proces in niveau's opgedeeld, waardoor het proces verder gedetailleerd wordt beschreven. Het terugkeren naar een hoger liggend niveau wordt aangegeven met een gebeurtenis die in een zwart bolletje met een cirkel omheen eindigt. Dit betekent dat de beschreven toestand in het hoger niveau verlaten wordt. Een pijl vanuit een zwart bolletje wijst de begintoestand aan.

5.2.1 Toestand diagram Sendcom

Hieronder (figuur 5.2 t/m 5.5) staan de toestand diagrammen die behoren bij het Sendcom proces.

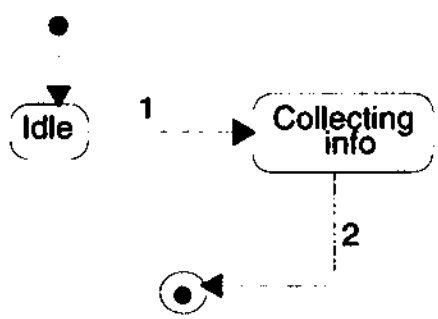


Figuur 5.2 Status diagram, Sendcom (top niveau)

1: data ontvangen
2: Recog_obj verzonden

In het diagram (figuur 5.2) is toestand "waiting for data" de begintoestand. In deze toestand kijkt het proces of er nieuwe objecten zijn. Bij het optreden van gebeurtenis 1, heeft Sendcom data (nieuw object die gedetecteerd is op het beeldscherm) ontvangen. Het proces verkeert dan in toestand "processing data". In deze toestand wordt de ontvangen data geschreven in het object *Recog_obj* en wordt vervolgens overgedragen naar het Receivecom proces. Na het overdragen van *Recog_obj* en het ontvangen van een bevestiging vanuit Receivecom, keert Sendcom terug naar de begintoestand.

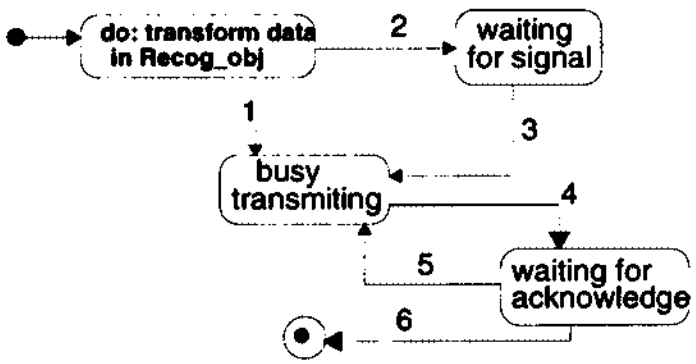
De toestanden "waiting for data" en "processing data" worden verder uitgewerkt in subdiagrammen. Deze zijn in figuur 5.3 respectievelijk figuur 5.4 weergegeven.



Figuur 5.3 Status diagram Waiting for data

1: Er is een verandering opgetreden
2: Data is ontvangen

In de "Idle" toestand wordt nagegaan of er nieuwe data aanwezig is. Er wordt dus steeds een polling uitgevoerd naar nieuwe data. Indien er nieuwe data is ontdekt, gaat het proces over tot inlezen van deze nieuwe informatie (toestand "Collecting info"). Nadat de informatie is ingelezen kan het proces overgaan tot het transformeren van deze informatie in *Recog_obj*. Dit gebeurt in toestand "Processing data" die hieronder wordt afgebeeld.

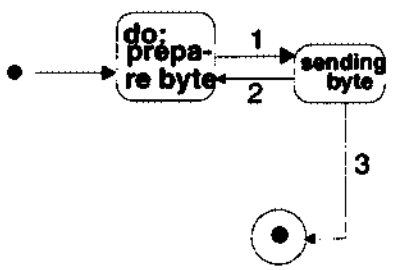


Figuur 5.4 Status diagram, processing data

1: versturen van Recog_obj [Receivecom is gereed]
2: [Receivecom niet gereed]/wacht op signaal
3: signaal ontvangen
4: klaar met verzending
5: hertransmissie van Recog_obj [TIME OUT]
6: bevestiging ontvangen

In de begintoestand wordt de data getransformeerd in *Recog_obj*. Na de transformatie gaat het proces over tot verzending van *Recog_obj*, indien Receivecom gereed is voor ontvangst, anders gaat het proces in de wacht toestand. Na het verzenden van *Recog_obj* wacht het proces op een bevestiging. Indien er geen bevestiging is ontvangen (TIME OUT), begint het proces met hertransmissie van *Recog_obj*. Ontvangt Sendcom een bevestiging, dan kan die opnieuw pollen voor informatie.

De toestand "busy transmitting" in figuur 5.4 wordt ook verder uitgewerkt in een subdiagram, zie figuur 5.5.



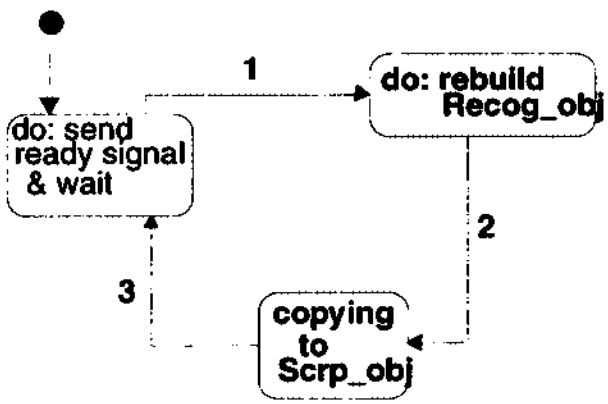
Figuur 5.5 Status diagram, busy transmitting

1: byte verzenden
2: byte is verzonden
3: laatste byte is verzonden

Elke byte die verzonden moet worden, wordt ingepakt voor de transmissie. Na het versturen van een byte (gebeurtenis 2), wordt het volgende byte gereed gemaakt voor verzending. Wanneer de laatste byte verstuurd wordt (gebeurtenis 3), gaat het proces wachten op een bevestiging (dit zie je in het diagram een niveau hoger figuur 5.4).

5.2.2 Toestand diagram Receivecom

Het toestand diagram voor het Receivecom proces is in figuur 5.6 afgebeeld. Het diagram beschrijft het eerder beschreven scenario (paragraaf 5.1.2) voor dit proces.



Figuur 5.6 Toestand diagram Receivecom

1: **Eerste byte ontvangen**
2: **Kopieer object in Scratch pad**
3: **Object gekopieerd/ready signaal (bevestiging) sturen**

Receivecom stuurt een signaal naar Sendcom, zodat deze weet dat Receivecom gereed is voor ontvangst. Bij het optreden van gebeurtenis 1 gaat het proces over tot ontvangst van de bytes waarna *Recog_obj* opnieuw wordt samengesteld zodat het in *Scrp_obj* (Scratch Pad) gekopieerd kan worden (gebeurtenis 2). Na het kopieren (gebeurtenis 3) wordt er weer een bevestiging signaal gestuurd waardoor Sendcom weet dat het object goed is aangekomen.

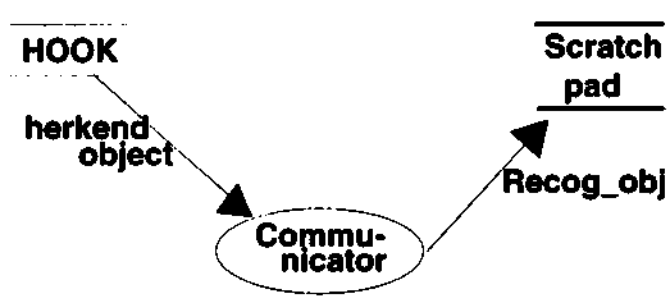
Nu het dynamisch model is opgesteld kan het functioneel model heel eenvoudig tot stand komen, omdat we al een idee hebben welke operaties het systeem op de gevonden objecten uitvoert. Door het functioneel model op te stellen krijg je een beter overzicht van deze operaties. Het functioneel model wordt in hoofdstuk 7 beschreven.

6 Functioneel model

Het functioneel model beschrijft hoe de gegevens door het systeem verwerkt worden zonder rekening te houden met de volgorde van verwerking, de beslissingen en/of de structuur van de objecten. Het functioneel model laat zien hoe deze gegevens van elkaar afhankelijk zijn. Deze functionele afhankelijkheid wordt in de vorm van data flow diagrammen geïllustreerd. Na het creëren van het functioneel model is het duidelijk welke operaties (functies) het systeem op de objecten uitvoert. Voor het opstellen van de data flow diagrammen is het noodzakelijk om te weten wat de invoer en uitvoer van het systeem zijn. Dit geeft een startpunt voor het opstellen van de data flow diagrammen,.

6.1 Identificatie van input en output

Voor het systeem hier zijn de input en output inmiddels al bekend vanuit de documenten hiervoor. Voor duidelijkheid wordt dit in figuur 6.1 nog eens afgebeeld in de vorm van een context diagram. In diverse tekeningen hieronder is Scratch Pad tussen twee horizontale strepen gezet, dit geeft aan dat Scratch Pad een opslag bestand is. Een vierkante symbool duidt eenextern systeem aan.

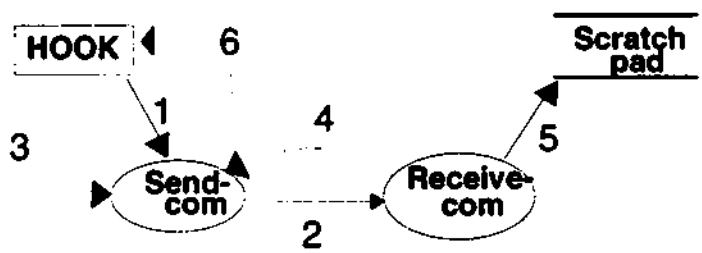


Figuur 6.1 Input/Output van het Communicator systeem

In het diagram zie je van wie het systeem de input (herkend object) ontvangt en aan wie de output (Recog_obj) verstrekt wordt. De data flow diagrammen die hierna volgen beschrijven hoe de input verwerkt wordt tot de output.

6.2 Data flow diagram

Aangezien dat het systeem gesplitst is in twee deelprocessen, wordt het data flow diagram voor beide deelprocessen apart beschreven. In figuur 6.2 wordt deze splitsing afgebeeld.



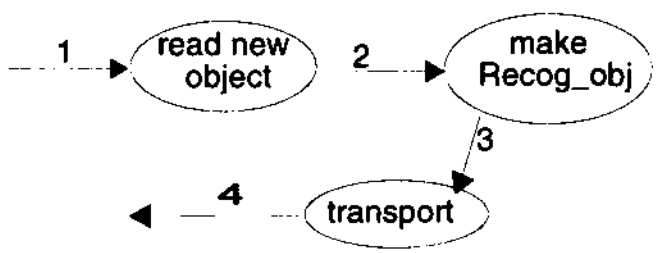
Figuur 6.2 Top niveau data flow diagram

Communicator.

1: **herkend_object**
2: **bytes**
3: **answer_to_poll**
4: **bevestiging**
5: **Recog_obj**
6: **poll**

Wat Sendcom en Receivecom doen wordt hier verder niet behandeld, omdat deze in het dynamisch model uitvoerig is uitgelegd (paragrafen 5.1.1 en 5.2.1). Voor beide processen wordt het data flow diagram verder uitgediept in de hierna volgende paragrafen.

6.2.1 Data flow diagram Sendcom

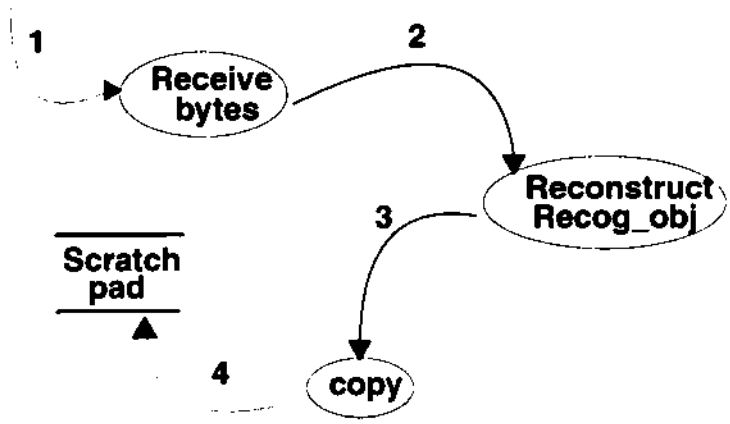


Figuur 6.3 Sendcom data flow diagram

1: herkend object
2: Idem aan 1
3: Recog_obj
4: bytes

"Read new object" zorgt voor het inlezen van het object dat herkend is. Na het inlezen van het object wordt die aan de functie "make Recog_obj" overgedragen, zodat het object *Recog_obj* gemaakt kan worden. *Recog_obj* kan nu door "transport" functie verstuurd worden naar Receivecom.

6.2.2 Data flow diagram Receivecom



figuur 6.4 Data flow diagram Receivecom

1: bytes
2: alle ontvangen bytes
3: Recog_obj
4: Recog_obj

"Receive bytes" zorgt voor de ontvangst van een reeks karakters (bytes) waarna deze reeks bytes weer omgezet kan worden in *Recog_obj* door "Reconstruct Recog_obj". "Copy" zorgt ervoor dat *Recog_obj* gekopieerd wordt in *Scratch Pad.*

6.3 Functie beschrijving

Nu dat de operaties (functies) bekend zijn kun je een grove beschrijving van de implementatie opzetten in de vorm van een pseudocode. De operaties die op de objecten worden uitgevoerd, worden hieronder als functies opgenomen. Hier wordt het nadruk gelegd op wat de functies doen en niet hoe het gedaan wordt.

Sendcom
input : bevestiging, answer_to_poll, herkend_object
output: bytes
{ Indien bevestiging ontvangen van Receivecom
Pollen totdat er een positief antwoord binnenkomt
if (answer_to_poll = True)
{ Nieuw object inlezen
Nieuw object transformeren in Recog_obj
Recog_obj byte voor byte sturen naar Receivecom
Wacht op bevestiging
}
}

Receivecom
input : bytes
output: bevestiging, Recog_obj
{ Een bevestiging naar Sendcom sturen bij opstart
Ontvang bytes die binnenkomen
Reconstrueer Recog_obj
Kopieer Recog_obj in Scratch pad
Stuur een bevestiging naar Sendcom
}

Read_new_object
input : ---
output:: herkend_object
{ Kopieert het nieuw object dat herkend is door HOOK }

Make_Recog_obj
input : herkend_object
output: Recog_obj
{ Deze functie transformeert het herkend object in Recog_obj }

Transport
input : Recog_obj
output: bytes
{ Neem een byte
Maak byte gereed
Stuur byte vie seriele poort
}

Receive_bytes
input : byte
output: reeks bytes
{ Ontvangt een reeks bytes
Geeft deze reeks bytes door
}

Reconstruct_Recog_obj
input : reeks bytes
output: Recog_obj
{ Reconstrueer Recog_obj uit de reeks bytes }

Copy
input : Recog_obj
output: ---
{ Kopieert het object Recog_obj in de Scratch pad }

Wat bij deze functies nog missen zijn de computer code van een of andere programmeertaal. Dit vindt plaats bij de implementatie van het hier ontworpen systeem.

7 Conclusies

Je hebt hiervoor gezien hoe door gebruik te maken van een object georiënteerd modelleringstechniek (m.n. OMT), een systeem is ontwikkeld. Eerst wordt een analyse gedaan van de situatie, daarna wordt een objecten- dynamisch- en functioneel model opgesteld die het systeem gedetailleerd beschrijven. Aan de hand van dit ontwerp is vervolgens overgegaan tot implementatie. Wat er verder bereikt is en hoe het verder zou kunnen gaan wordt in dit hoofdstuk beschreven.

7.1 Wat werd bereikt ?

De opdracht was: het integreren van de reeds in het EU-project ontwikkelde GUI aan de SoundTablet software inclusief aanpassingen en optimalisatie die betrekking hebben op zowel de methoden voor ophalen van GUI informatie als het simuleren van muis acties en toetsenbord acties naar de GUI.

Uit de opdracht beschrijving kun je afleiden dat de opdracht uit twee delen bestaat. Het eerste gedeelte heeft betrekking op de conversie en transport van GUI informatie en het tweede gedeelte heeft betrekking op het simuleren van muis- en toetsenbordacties. Het systeem dat ontworpen is doet het volgende:

- het systeem zorgt voor het ophalen van de GUI informatie,
- het omzetten van deze informatie zodat de SoundTablet software deze kan begrijpen en
- voor het versturen van de omgezette informatie via de seriële poort naar de machine waarop de Soundtablet software draait zodat deze die informatie kan gebruiken.

Vanwege tijd te kort is er maar een deel van het ontwerp geïmplementeerd. Dat deel dat voor het opsturen en ontvangen van de omgezette informatie is geïmplementeerd (zie bijlage). Meer tijd is gaan zitten in het bestuderen van OMT, voordat begonnen werd met de opdracht (OMT was hiervoor niet bekend). Weinig ervaring bij ontwikkeling van WINDOWS applicatie, onduidelijkheden m.b.t. HOOK: deze was niet goed gedocumenteerd. Dit alles heeft onverwacht veel tijd gekost.

Een voorwaarde bij deze opdracht was dat het systeem met behulp van OMT ontwikkeld moest worden. Dit is de methode die men intern gebruikt voor software ontwikkeling. Dit is een degelijke methode dat gebruikt wordt bij object georiënteerd ontwerpen. Met deze methode kun je een ingewikkeld systeem heel systematisch ontwerpen zonder dat je overzicht over het geheel verliest. Een sterk punt van deze methode is, dat als een ontwerper van een systeem OMT heeft gebruikt, het ontwerp aan iemand anders (die bekend is met OMT) kan geven om het in een programmeertaal te implementeren. Het ontwerp is dus helemaal taalonafhankelijk.

Een andere voorwaarde was dat het systeem in Borland C++ (een deel in BC++ voor Windows) geïmplementeerd moest worden. Doordat C++ een object georiënteerd taal is, is het vrij eenvoudig om een object georiënteerd ontwerp in C++ te vertalen. Alle drie modellen (object- dynamisch en functioneel moel) kunnen direct vertaald worden in C++ code. Het objecten model bevat een groot deel van de declaratie structuur: de specificatie van de klassen, attributen en associaties. Het dynamisch model beschrijft de besturing strategie van het systeem: procedureel- of gebeurtenis gestuurd of multi-tasking. Het functioneel model, dat de functionaliteit voor zich neemt. C++ biedt de mogelijkheid om object klassen, operaties (functies) op deze objecten, de implementatie van deze operaties en de applicatie apart te definiëren (in aparte files die geinclude kunnen worden). Op deze manier hou je in je implementatie dezelfde structuur als in je ontwerp. Door toekomstige aanpassingen eerst in het ontwerp op te nemen, weet je precies waar in de implementatie die aanpassingen aangebracht moeten worden.

7.2 Suggesties

Met het ontwerp kan de rest van het systeem geïmplementeerd worden. Dat deel dat betrekking heeft op het ophalen van de beeldscherm informatie en het TIME OUT mechanisme, moet nog geïmplementeerd worden, voordat dit ontwerp volledig is geïmplementeerd. Hierna kan het ander gedeelte van de opdracht uitgevoerd worden (simuleren van muis- en toetsenbordacties). Daarnaast kan het systeem in bepaalde opzichten verbeterd worden. Sendcom kan bijvoorbeeld een sliding window kunnen gebruiken die een x aantal objecten (Recog_obj) die opgestuurd zijn naar Receivecom, vastlegt. Bij gebruik van deze sliding window kan Sendcom voor nieuwe informatie gaan pollen terwijl die op een bevestiging vanuit Receivecom wacht. Er kan ook een kopie van Scratch Pad bij Sendcom bijgehouden worden: indien er op een of andere manier de informatie in de Scratch Pad op pc2 verloren gaat, kan Receivecom aan Sendcom vragen om de informatie in de kopie op te sturen zodat de CSM het beeld kan herbouwen.

Deze aanpassingen en verbeteringen kunnen leiden tot nieuwe eisen waaraan het systeem moet voldoen met als gevolg dat dit document aan de nodige aanpassingen zal ondergaan. Dit is het voordeel van software engineering methoden.

Bijlage

Een onderdeel van de implementatie

Objecten declaratie en operatie code
Proces Sendcom

```
*************************************************************************

#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

#define TRUE    1
#define FALSE   0

char *str;
DCB my_dcb;

// Object klassen
struct coord
        { int xcord,
              ycord;
        };

struct Recog_obj
        { char  *obj_name;
          coord upperleft,
                bottomright;
          char  *Tekst;
          int   Tekstcolor;
          void Transport(int Port);
        };

struct Scrp_obj
        { Recog_obj newobject;
          int     *Next,
                  *Prev;
        };

//====================

int Initialiseren()
{  int err,temp;

   temp= OpenComm("COM1",1024,2);
   err= BuildCommDCB("COM1:1200,n,7,1", &my_dcb);
   err= SetCommState(&my_dcb);
   return temp;
}
```

```
//=======================

int send_string(int Port)
{  int i= 0, re= TRUE, succes;

   do
         { // Neem een byte en stuur byte op lijn
          succes= TransmitCommChar(Port, str[i]);
          if ( !succes )
                        i++;
         } while ( str[i] != '\x0' );
   while ( TransmitCommChar(Port, str[i]));
   return re;
}

//====================

void Receive_ACK(int Port)
{  void FAR* ACK[2];
   int n_b;

   do
         n_b= ReadComm(Port, ACK, 2);
   while( n_b <= 0 );
}

//===============

void Recog_obj::Transport(int Port)
{  int  x, y, col, st;

   // object_name opsturen
   str= obj_name;
   st= send_string(Port);
   Receive_ACK(Port);

   // Upperleft opsturen
   x= upperleft.xcord; y= upperleft.ycord;
   str= itoa(x, str, 10);
   st= send_string(Port);
   Receive_ACK(Port);

   str= itoa(y, str, 10);
   st= send_string(Port);
   Receive_ACK(Port);
```

```
    // Bottomright opsturen
    x= bottomright.xcord; y= bottomright.ycord;
    str= itoa(x, str, 10);
    st= send_string(Port);
    Receive_ACK(Port);

    str= itoa(y, str, 10);
    st= send_string(Port);
    Receive_ACK(Port);

    // Tekst opsturen
    str= Tekst;
    st= send_string(Port);
    Receive_ACK(Port);

    // zet tekst_color om in string
    col= Tekstcolor;
    str= itoa(Tekstcolor, str, 10);
    st= send_string(Port);
    Receive_ACK(Port);
}

//====================


//int main()
WINAPI WinMain(HINSTANCE hInstance,
                        HINSTANCE hPrevInstance,
                        LPSTR lpCmdLine, int nCmdShow)
{  Recog_obj  R_object;
    char t;
    int  Port, init;

    Port= Initialiseren();
    R_object.obj_name= "Window";
    R_object.upperleft.xcord= 20;
    R_object.upperleft.ycord= 10;
    R_object.bottomright.xcord= 40;
    R_object.bottomright.ycord= 30;
    R_object.Tekst= "Dit is een test voor GUI";
    R_object.Tekstcolor= 2;

    R_object.Transport(Port);
    return 0;
}
```

objecten declaratie en operatie code
proces Receivecom

```
*****************************************************************************

#include <stdlib.h>
#include <stdio.h>
#include <rt_kerne.h>
#include <conio.h>

#define FALSE 0
#define DIGICOM  1        // 0 = COM1   1=COM2
#define BITNUMBER 7
#define PARITY_TYPE 2    // 0 = EVEN   1=ODD   2=NONE
#define STOP_BITS 1
#define BAUDRATE 1200
#define TRUE 1

// OBJECT KLASSEN
struct coord
                { int xcord,
                        ycord;
                };

struct Recog_obj
                { char  *obj_name;
                    coord upperleft,
                                    bottomright;
                    char  *Tekst;
                    int   Tekstcolor;
                    void Herbouw();
                };

struct Scrp_obj
                { Recog_obj newobject;
                     int      *Next,
                                                *Prev;
//                  void Kopy_object(Recog_obj object);
                };

RTKWindow *MainWin;
char str[1024];
int i, k= 0;
Scrp_obj Linked_list[1024];
```

```
//====================

void Rec_byte()
{   char out,inbuf;

   //initialise the communication port
   InitPort(DIGICOM,BAUDRATE,PARITY_TYPE,STOP_BITS,BITNUMBER);

   //enable hardwareinterrupts
   EnableCOMInterrupt(DIGICOM,16);
//  Wprintf(MainWin,"\r\n");
   do
        { RTKGet(ReceiveBuffer[DIGICOM],&inbuf);
             out = (inbuf&0x007F);
             str[i]= out;   // (inbuf & 0x007F);
             i++;
        }
   while( str[i-1] != '\x0' );
}

//====================

void Receive_string()
{  Rec_byte(); }

//====================

void Recog_obj::Herbouw()
{  char bevest= '1';

   i= 0;
   Receive_string();
   obj_name= str;
   printf("obj_name= %s\n", obj_name);
   SendChar(DIGICOM, bevest);

   i= 0;
   Receive_string();
   upperleft.xcord= atoi(str);
   printf("upperleft.xcord= %d\n", upperleft.xcord);
   SendChar(DIGICOM, bevest);

   i= 0;
   Receive_string();
   upperleft.ycord= atoi(str);
   printf("upperleft.ycord= %d\n", upperleft.ycord);
```

```
    SendChar(DIGICOM, bevest);

    i= 0;
    Receive_string();
    bottomright.xcord= atoi(str);
    printf("bottomright.xcord= %d\n", bottomright.xcord);
    SendChar(DIGICOM, bevest);

    i= 0;
    Receive_string();
    bottomright.ycord= atoi(str);
    printf("bottomright.ycord= %d\n", bottomright.ycord);
    SendChar(DIGICOM, bevest);

    i= 0;
    Receive_string();
    Tekst= str;
    printf("Tekst= %s\n", Tekst);
    SendChar(DIGICOM, bevest);

    i= 0;
    Receive_string();
    Tekstcolor= atoi(str);
    printf("Tekstcolor= %d\n", Tekstcolor);
    SendChar(DIGICOM, bevest);
}

//====================

void Kopy_object(Recog_obj *object)
{ if ( k==0 )
        { Linked_list[k].newobject= *object;
            Linked_list[k].Prev= NULL;
            Linked_list[k].Next= NULL;
        }
    else
        { Linked_list[k].newobject= *object;
            Linked_list[k].Next= NULL;
//          Linked_list[k].Prev= &Linked_list[k-1];
//          Linked_list[k-1].Next= &Linked_list[k];
        }
    k++;
}
```

```
//====================

int main()
{  Recog_obj R_object;

   clrscr();
   RTKernelInit(3);
   RTKeybrdInit();
   RTTextIOInit();
   RTComInit();

   R_object.Herbouw();
   Kopy_object(&R_object);
   return 0;
}
```

Literatuur

Edwards A.D.N.
1987, "Adapting user interfaces for visually disabled users", *Doctoral thesis, Open universiteit van Milton Keynes.*

Gerrits A.H.J., Poll L.H.D. en Waterham R.P.
1993, "VISA Comp current screen module", *software engineering document.*

Poll L.H.D. en Waterham R.P.
1995, "Graphical User Interfaces and Visually Disabled Users", *IEEE Transactionons rehabilitation engineering vol. 3.*

Rumbaugh J., Blaha M., Premerlani W., Eddy F. en Lorensen W.
1991, "Object Oriented Modeling and Design"