In hoeverre kunnen de huidige GUI-tools ontwikkelaars helpen? Waar moeten zij op letten bij de selectie van hulpmiddelen voor GUI-ontwikkeling? Dit artikel brengt een structurering aan in de wirwar van verschillende soorten tools en geeft een aantal aandachtspunten voor het kiezen van tools. Daarbij komen ook de huidige stand van zaken en wensen voor de toekomst aan bod.

thema

Groeiend aantal platforms zorgt voor complexere GUI's

Tools volgen (te) traag

Om een goede interface te kunnen ontwikkelen, heeft de organisatie allereerst een software-ontwikkelproces nodig dat voldoende aandacht schenkt aan de gebruikersinterface gedurende het gehele traject. Vervolgens kan zij het ontwikkelproces met GUI-tools ondersteunen. Wat doen GUI-tools voor de ontwikkelaar? Ze helpen bij:

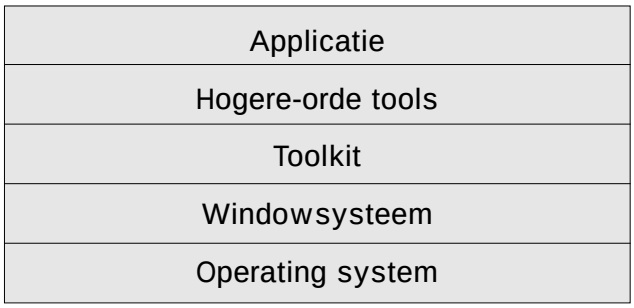
- het ontwerp van een GUI op basis van een beschrijving van de taken van de gebruikers;
- de implementatie van een GUI op basis van een specificatie van het ontwerp;
- het evalueren van de GUI na het ontwerp.

Verder helpen ze bij het maken van makkelijk te gebruiken interfaces, dragen ze bij aan consistentie in de gebruikersinterface voor verschillende applicaties en vereenvoudigen ze portabiliteit. De ontwikkelaars kunnen (ook als niet-programmeur) een GUI snel prototypen en bouwen om alternatieve ontwerpen te onderzoeken. Ze kunnen specificaties presenteren en valideren en eenvoudig veranderingen aanbrengen. Samenwerking tussen ontwikkelaars gaat gemakkelijker, ze hoeven minder code zelf te schrijven (wat leidt tot hogere betrouwbaarheid) en ontwikkelaars kunnen eenvoudiger de scheiding tussen GUI en applicatie aanbrengen.

TOOLSPECTRUM GUI-software bestaat uit verschillende lagen (zie de afbeelding). Op het operating system draait een windowsysteem (bijvoorbeeld X-Windows of Microsoft Windows). Een toolkit is vaak nauw verweven met het windowsysteem en biedt een bibliotheek van interface-elementen (vaak widgets genoemd, denk aan een tekstinvoerveld of een knop), waar een gebruikersinterface gebruik van maakt. Voorbeelden zijn Motif en OpenLook (voor X-Windows), elk met hun eigen 'look and feel'. Softwarepakketten kunnen verscheidene lagen bestrijken. Zo is Microsoft Windows 98 naast een windowsysteem ook een toolkit en verweven met het operating system MS-DOS.

Om het programmeren van gebruikersinterface van een applicatie (het bovenste niveau in de afbeelding) op basis van een toolkit eenvoudiger te maken, bestaan er hogere-orde tools. Er zijn honderden van deze tools. Deze tools dragen bij aan de GUI tijdens het ontwikkeltraject. Wanneer ze een significante bijdrage leveren aan de executerende GUI, worden ze vaak User Interface Management System (UIMS) genoemd. Hogere-orde GUI-tools gebruiken verschillende stijlen van specificatie:

- **Taalgebaseerd**
  Een gespecialiseerde taal specificeert de interface. Hieronder vallen state transition networks (bijvoorbeeld VAPS), context-vrije grammatica's (Yacc/Lex), eventgebaseerde talen (HyperTalk), declaratieve talen (Motif UIL) en constrainttalen (C32), screen scrapers (Easel), database interfaces (een 4GL als Oracle) en visuele programmeeromgevingen (Visual Basic, Delphi).
- **Applicatieraamwerken**
  De ontwikkelaar kan een klassehiërarchie van interface-objecten die het tool beschikbaar stelt, aanpassen. Het raamwerk regelt de layout en de controle en biedt vaak meer dan de toolkit. Denk aan tools als MacApp en de Microsoft Foundation Classes (MFC).
- **Specificatie voor automatische generatie**
  Het tool genereert code die complexe plaatsing, formattering en ontwerp van de gebruikersinterface regelt, gebaseerd op een hogere-orde specificatie. Een voorbeeld is Mastermind.



AFBEELDING: Lagen van GUI-software

- **Directe grafische specificatie**
  De ontwikkelaar kan met deze tools objecten op het scherm plaatsen (met bijvoorbeeld de muis) en op die manier ten minste een deel van de interface specificeren. Interface builders (bijvoorbeeld NeXT Interface Builder, X-Designer) vallen in deze categorie.

In de praktijk gebruiken geïntegreerde ontwikkelomgevingen verscheidene stijlen van specificatie. Zo is Borland Delphi een visuele programmeeromgeving met een interface builder en biedt het een applicatieraamwerk.

Verder bestaan er tools die gelieerd zijn aan het maken van een GUI, bijvoorbeeld tools voor het maken van helpfaciliteiten en tutorials, programma's om iconen mee te maken en bibliotheken van icons, etc.

Naast hogere-orde tools waarbij het voornamelijk draait om de implementatie van een gebruikersinterface, bestaan er ook tools voor de andere stadia van het proces van interface-ontwerp: analyse, ontwerp en testen. Voorbeelden zijn een analysetool, een style checker die controleert of de GUI voldoet aan de style guide van het platform en een bibliotheek van interactiepatronen die ontwikkelaars kunnen raadplegen voor terugkerende vraagstukken. Bedenk ook dat er bij het ontwikkelen van een gebruikersinterface behalve programmeurs ook mensen in de rol van projectleider, informatie-analist, grafisch ontwerper en tekstschrijver zijn betrokken. Hun gezichtspunten zijn van belang bij het kiezen van GUI-tools. Verder zijn de eigenschappen van de eindgebruikers van belang voor de keuze. Het aantal eindgebruikers bepaalt bijvoorbeeld de keuze voor een lichte of zware tool.

Hierna ligt het accent op hogere-orde tools waarmee ontwikkelaars een GUI kunnen implementeren op het Microsoft Windows-platform (9X en NT) en het browserplatform. Het browserplatform behelst interfaces die via een World Wide Web-browser met de gebruiker communiceren. Op dit platform gebruiken de ontwikkelaars HTML en eventueel DHTML en JavaScript. Ook kunnen zij ActiveX-controls, Java-applets en browser plug-ins gebruiken om aanvullende interface-elementen te realiseren (of om met een server te communiceren). In essentie kan het ontwikkelteam de complete interface als ActiveX-component, Java-applet of met gebruik van plug-ins in de browser (bijvoorbeeld voor ShockWave Flash) realiseren, maar eigenlijk is dat een Windows-interface binnen de browser.

CRITERIA Het eerste waar men aan denkt bij de selectie of evaluatie van een GUI-tool zijn prijs, de platforms waarop het draait en waarvoor een GUI ontwikkeld kan worden, de feature list en leveranciersondersteuning. Verder is de programmeer- of specificatietaal die het gebruikt van belang, omdat in een organisatie vaak al veel ervaring is met bepaalde talen en vaak veel bestaande software in een bepaalde taal geschreven is.

De overwegingen hieronder verleggen echter de aandacht van uiterlijke en technische eigenschappen van tools naar de gewenste rol van GUI-tools in het proces van software-ontwikkeling.

Allereerst doet zich de vraag voor: waarom wil men een tool en waarvoor wil men het precies gebruiken? Wat is de gewenste rol in het GUI-ontwikkeltraject (analyse, ontwerp, implementatie, testen, evaluatie)? Dit moet ook bezien worden vanuit het ontwikkeltraject van de gehele applicatie. Er zijn tools waarin GUI-ontwikkeling slechts een onderdeel is en er zijn specifieke GUI-tools. Als het tool het hart van het ontwikkeltraject vormt, is het belangrijk dat het de gemaakte modellen consistent houdt. Als het tool slechts voor GUI-ontwikkeling gebruikt wordt, is de integratie met andere tools van belang. Een praktijkvoorbeeld is een ontwikkelteam dat de modellering van de GUI in de presentatielaag consistent wil houden met het objectmodel in de applicatielaag en het datamodel in de persistentielaag. Merk op dat code ook als een model gezien kan worden. De mogelijkheden voor debugging daarvan kunnen van belang zijn. Verder is een organisatie geïnteresseerd in tools met hoge ontwikkelsnelheid die GUI's van voldoende kwaliteit leveren.

Bekende ontwikkelomgevingen (zoals Microsoft Visual InterDev, Microsoft Visual Basic of Borland Delphi) verschaffen vaak een programmeeromgeving waarmee een ontwikkelaar zowel vanuit een taalspecificatie (respectievelijk HTML, Basic en Object Pascal) als vanuit een GUI-builder een gebruikersinterface kan maken. Vaak genereert de GUI-builder een taalspecificatie. Deze code is vaak onbegrijpelijk (tenzij het tool dit netjes verbergt zoals Delphi doet) en men moet zeker oppassen met het aanpassen ervan. Dat laatste is helaas vaak nodig, omdat de mogelijkheden van een GUI-builder beperkt zijn ten opzichte van de mogelijkheden die de taal aanbiedt. Ze werken vaak met absolute (in tegenstelling tot relatieve) positionering van interface-elementen. Grafische editors voor HTML werken vaak wel met relatieve positionering en houden code (HTML) en grafische specificatie consistent.

Een vraag bij absolute positionering is: wat gebeurt er als de interface van grootte moet veranderen? Kan de ontwikkelaar aangeven welke interface-elementen meeschalen? De NeXT Interface Builder was het eerste tool waarmee een ontwikkelaar dit volledig grafisch kon specificeren. Tegenwoordig kan dit met tools als Symantec VisualCafé en Forte for Java ook, zij het slechts gedeeltelijk. De ontwikkelaar moet de relatieve plaatsing en verhoudingen nog steeds op een niet grafische manier aangeven. Daarom gebruiken ontwikkelaars een GUI-builder in de praktijk meestal slechts voor eenvoudige, statische panelen.

PLATFORMS EN PORTABILITEIT Over platforms en portabiliteit in dit verband het volgende. Men zou verwachten dat in de loop van de tijd het aantal platforms vermindert, echter, er komen telkens nieuwe bij. Na de Windows-platforms (Microsoft Windows 95/98/NT 4.0, Apple Macintosh, OSF/Motif) kwam het browserplatform

op en momenteel hebben de hand held-platforms als PDA's (personal digital assistants) en mobiele telefoons de aandacht. Als platforms qua technische faciliteiten vergelijkbaar zijn, wil de ontwikkelaar er geen rekening mee houden dat de platforms hun beperkingen hebben. Dat zouden de tools moeten opvangen. Een bekende beperking van het browserplatform is bijvoorbeeld de afwezigheid van toestand in HTTP en het hand held-platform maakt gebruik van andere invoermogelijkheden (denk aan handbewegingen met de pen). Toegankelijkheid is een vraagstuk op zich: hoe ondersteunt het tool ons bij het maken van een interface die ook geschikt is voor mensen met een handicap (denk aan alternatieven voor het toetsenbord zoals spraakherkenning)?

Als de technische verschillen tussen platforms erg groot zijn, zal de ontwikkelaar daar wel rekening mee moeten houden. De hoeveelheid informatie die een platform kan weergeven speelt hierbij een belangrijke rol. De kleine schermen en het beperkt aantal beschikbare kleuren van een hand held vragen bijvoorbeeld om het weergeven van compactere informatie in vergelijking met een webbrowser. Een tool moet in dit geval assisteren in het onderling qua functionaliteit consistent houden van verscheidene GUI's.

Nu de interface-elementen voor kantoorapplicaties onderhand zijn uitgekristalliseerd, zou de ontwikkelaar portabiliteit cadeau willen krijgen als de interface verder geen bijzondere eisen stelt. GUI-tools gebruiken vaak een 'grootste gemene deler'-benadering: van de verschillende platforms kan de ontwikkelaar slechts de gemeenschappelijke elementen gebruiken, zoals in de Abstract Windowing Toolkit (AWT) van Java. Aan de andere kant wil de ontwikkelaar graag gebruik maken van de specifieke mogelijkheden van het platform. Een gebruiker van Microsoft Windows verwacht bijvoorbeeld ondersteuning van OLE (Object Linking and Embedding), iets wat door de 'grootste gemene deler'-benadering van de AWT moeilijk is. Ondersteuning van drag-and-drop tussen een Java-applicatie en een niet in Java geschreven applicatie is lang een punt geweest. Idealiter zou de ontwikkelaar de gebruikersinterface willen specificeren op een platformonafhankelijke manier (die portable is) en vervolgens voor de verschillende platforms de specifieke zaken aanbrengen. Nieuwe generatie-ontwikkelomgevingen (zoals The EDCubed Tool) bieden de mogelijkheid om een GUI-specificatie te vertalen naar een browser-interface (eventueel met gebruik van Java-applets) en naar een Windows-interface.

NIEUWE FUNCTIONALITEIT De mogelijkheden die platforms bieden, worden ook steeds groter. Waar vroeger een eenvoudige interface met invulformulieren volstond, komt nu meer vraag naar grafische elementen, animatie, 3D-visualisaties en samenwerken op een gedistribueerd systeem. Nieuwe mogelijkheden worden vaak in eerste instantie als component beschikbaar gesteld of als een (klasse) bibliotheek in het platform ondergebracht (vaak als API), die de gebruiker vervolgens als deel van het applicatieraamwerk kan gebruiken. Toolondersteuning komt pas later. Als de ontwikkelaar momenteel de genoemde elementen in de interface wil gebruiken, zal dat hem veel werk kosten, als er überhaupt al een bibliotheek voor zijn platform is. Dit probleem wordt alleen maar erger aangezien er geen eind aan de nieuwe ontwikkelingen in zicht is en de hoeveelheden te manipuleren informatie toenemen.

Voor een multi-mediabedrijf schieten tools vaak tekort en werken ze te beperkend. Sommige bedrijven maken daarom (van de grond af) hun eigen applicatieraamwerk, waarin genoemde elementen mogelijk zijn. Een creatieve vormgever kan bijvoorbeeld ronde panelen bedenken. De tools laten de ontwikkelaar dan aan zijn lot over. Een

Software-architectuur

De populaire drie-lagenarchitectuur is een voorbeeld van een standaardarchitectuur voor een applicatie. Een GUI-tool kan ondersteuning bieden bij het toepassen van een dergelijke architectuur, bijvoorbeeld door een raamwerk voor de applicatie te verschaffen. Stand-alone en client/server zijn de meest in tools voorkomende applicatie-architecturen, maar die zijn niet altijd gewenst. Ondersteunt het tool bijvoorbeeld ook andere architecturen, zoals het gebruik van gedistribueerde objecten (bijvoorbeeld via Enterprise JavaBeans)? Specifiek voor de GUI zal men vaak een model-view-controller-architectuur gebruiken. SmallTalk-ontwikkelomgevingen bevorderen het gebruik van deze architectuur al sinds jaar en dag. Ook kan het tool het gebruik van een eigen architectuur of raamwerkprogrammatuur ondersteunen. Belangrijk in elke architectuur is dat het de GUI-laag scheidt van de rest van de applicatie, om verscheidene views binnen de interface consistent te houden en portabiliteit te vereenvoudigen. GUI en applicatie raken vaak nauw verweven door de opzet van toolkits. Deze maken gebruik van aanroepen ('callbacks') naar de applicatie voor vrijwel elke gebeurtenis in de gebruikersinterface. Verder leidt het gebruik van databasecomponenten (zogenaamde data-aware componenten) in de GUI vaak tot inefficiëntie door verkeerde locking-strategieën, het overhalen van een gehele tabel terwijl dat niet nodig is en een slechte balans tussen client- en serververwerking. De scheiding tussen GUI en applicatie is meestal niet goed bij het gebruik van deze componenten. Daarnaast zou een scheiding van gedrag en verschijning wenselijk zijn. Mp3-spelers als WinAmp en Sonique bieden 'skins'. Deze specificeren niet alleen een paar bitmaps maar ook de positionering en vorm van interface-elementen, los van hun gedrag. In ontwikkeltools komt dit concept nog niet voor, hooguit in de 'pluggable look and feel' van het JFC/Swing applicatieraamwerk (een concept dat al bestond in SmallTalk-omgevingen). Deze breidt de AWT uit door een 'virtuele toolkit' aan te bieden die buiten de toolkit om interface-elementen en mechanismen verschaft. Deze werken daarom niet precies hetzelfde als de gebruiker van de toolkit gewend is.

belangrijke vraag bij de keuze van tools is dus in hoeverre de ontwikkelaar met het tool de volledige mogelijkheden van het platform kan benutten.

Voor klassieke kantoorapplicaties kan het ontwikkelteam het nog even zonder deze mogelijkheden doen en bieden de tools redelijke ondersteuning. Willen zij echter meer, dan laten de tools hen in de steek. Voor samenwerking zou bijvoorbeeld een NetMeeting component handig zijn, maar dat is niet genoeg. Het rekening houden met samenwerking heeft voor de gehele applicatie gevolgen, maar daar staat de ontwikkelaar alleen voor. Het Taligent applicatieraamwerk (dat nooit een commercieel succes is geworden) is een voorbeeld van een raamwerk dat een ontwikkelaar hulp biedt bij het maken van applicaties waarin samenwerking inherent is.

TOEKOMSTPERSPECTIEF Ontwikkelaars willen ondersteuning van de modellen, notaties en het ontwikkelproces van bekende GUI-ontwikkelmethoden. Denk bijvoorbeeld aan modellen uit methoden als Jacobson, die gebruik maken van taakscenario's of use-cases, aan het hiërarchisch taakmodel uit Hierarchical Task Analysis (HTA) of aan het objectmodel uit OMT. Een interessante vraag is of het tool het gedrag van een interface expliciet kan modelleren.

Zoals eerder gezegd is het van belang een software-ontwikkelproces te gebruiken dat voldoende aandacht schenkt aan de gebruikersinterface gedurende het gehele traject. Ook zijn er ontwikkelprocessen die zich specifiek richten op het GUI-ontwerp (zoals Studio, Guide en Effectief GUI-ontwerp). Naast ondersteuning van deze methoden wil het ontwikkelteam een eigen ontwikkelmethode kunnen gebruiken. Verder moeten zij bij toolselectie letten op de mogelijkheden voor het snel prototypen en iteratief ontwikkelen van een interface.

Een aantal tools maakt automatisch vertaalslagen tussen de verschillende modellen van de gebruikersinterface. Voorbeelden zijn generatie van code of van documentatie. Belangrijke vragen hierbij zijn welke modellen de ontwikkelaar moet invullen om te kunnen genereren en voor welke platforms het tool code kan genereren. De vierde-generatietalen (4GL's) bieden vaak mogelijkheden om een GUI te genereren vanuit het datamodel. Of dit ook leidt tot een GUI waar de gebruiker prettig mee werkt is zeer de vraag. Tools uit de onderzoekswereld die uit een abstracte specificatie van de inhoud en het gedrag van een gebruikersinterface automatisch een GUI genereren, zijn in de markt nooit een succes geworden. Toch heeft deze manier van specificatie een goed toekomstperspectief, gezien de immer complexer wordende interfaces en het toenemend aantal platforms.

OPENHEID Met het oog op het ontwikkeltraject en integratie met andere tools is de openheid van het tool van belang. Als het tool gebruik maakt van een eigen programmeer- of scripttaal (zoals bijvoorbeeld bij Uniface) of klassebibliotheek, moet het ontwikkelteam letten op de mogelijkheden daarvan. Vooral voor een gebruikersinterface schieten deze namelijk vaak tekort; wat te doen als een gewenste interface niet gerealiseerd kan worden binnen de taal? Ontwikkelaars moeten daarom letten op de mogelijkheden voor koppeling met andere tools, uitbreiding met bijvoorbeeld een API en portabiliteit naar een andere omgeving. Zo kan MacroMedia Director het eigen formaat exporteren naar ShockWave, bijvoorbeeld voor gebruik in een Webbrowser.

Verder zal het ontwikkelteam vaak gebruik maken van bestaande software. Het is van belang te weten welke modellen de ontwikkelaar daaruit kan verkrijgen (reverse engineering). Ook kan het tool ondersteuning bieden bij het inbedden van bestaande software in de nieuwe software, bijvoorbeeld via 'wrapping'. Een tool als Delphi realiseert dit door de mogelijkheid ActiveX-componenten of VBX-controls (een formaat afkomstig van Visual Basic) in te bedden in de gebruikersinterface. Java-tools gebruiken hiervoor JavaBeans.

GEBRUIKSASPECTEN De eenvoud en gebruiksvriendelijkheid van de gebruikersinterface van het tool zelf is een belangrijk criterium. Aangezien de te ontwikkelen GUI vaak het meest ingewikkelde deel van de applicatie is, is het belangrijk dat de gebruiker de complexiteit kan beheersen. Daarnaast ondersteunen sommige tools samenwerking tussen verschillende ontwikkelaars, bijvoorbeeld door een vorm van versie- en configuratiebeheer of mogelijkheden tot autorisatie te bieden.

Een manier om de productiviteit van de ontwikkelaar te verbeteren is het op maat maken van het tool door het in te stellen naar specifieke eigen wensen. In tools is echter nog geen ondersteuning van adaptieve interfaces, die zich aanpassen aan de gebruiker. Een andere manier waarop een tool de productiviteit verhoogt, is door hergebruik te faciliteren en te stimuleren. Hergebruik is alleen goed mogelijk als het weinig inspanning van de ontwikkelaars vraagt. Gebruikt het tool een repository om GUI-objecten en hun onderlinge relaties in op te slaan? Een vorm van hergebruik, zei het op een wat laag niveau, is dat tools vaak naast de toolkit een aantal uitgebreidere interface-elementen aanbiedt. Denk bijvoorbeeld aan een dialoog waarmee de gebruiker een vraag moet beantwoorden. Daarnaast zijn er wat hoger-niveau-componenten op de markt, bijvoorbeeld om grafieken weer te geven.

KWALITEIT Het tool kan de kwaliteit van de modellen en de uiteindelijke GUI verbeteren door automatische controle van consistentie. Ondersteunt de interface bijvoorbeeld alle taken uit het taakmodel? Ook kan het analyseren van modellen en het sturen van het model-

Lees verder op pagina 48

Vervolg van pagina 42

leren een goed hulpmiddel zijn voor kwaliteitsverbetering. Het tool zou kunnen ondersteunen bij de controle of de GUI voldoet aan de style guide van het platform of de ontwikkelaar sturen bij het maken van een GUI die hieraan voldoet. In de praktijk zijn er tools die de mogelijkheden van layout beperken. Tcl/Tk bepaalt bijvoorbeeld zelf de precieze layout naar aanleiding van specificaties in de trant van “deze knop wil ik naast deze lijst”. Veel GUI-bouwers maken gebruik van een ‘grid’ dat de mogelijkheden voor positionering beperkt. Verder helpen de hierboven genoemde herbruikbare componenten tot meer consistentie. Erg ver gaan de meeste commerciële tools hier echter niet in.

SELECTIETRAJECT Het gebruik van de evaluatiecriteria helpt de aandacht te verleggen van meer uiterlijke of technische eigenschappen van tools naar de gewenste rol van GUI-tools in het proces van software-ontwikkeling. Daarbij moet dus allereerst duidelijk zijn wat de context van het tool is, dat wil zeggen de gewenste rol van het tool binnen het organisatiespecifieke ontwikkelproces.

Verder moet duidelijk zijn wat de visie van een organisatie op het gebruik van software-architectuur en standaard ontwerpoplossingen binnen software-ontwikkeling is. Pas als duidelijk is hoe bijvoorbeeld het gebruik van standaardcomponenten of raamwerken past binnen het organisatiespecifieke ontwikkelproces, kan het ontwikkelteam de geschetste criteria aanscherpen tot concrete eisen voor automatische ondersteuning. Ten slotte volgt de eigenlijke selectie of evaluatie van tools.

NIEUWE UITDAGINGEN De geschetste aandachtspunten voor het kiezen van GUI-tools bieden een organisatie houvast bij het selecteren van ontwikkelhulpmiddelen. De huidige tools zijn (nog steeds) primair gericht op het maken van gebruikersinterfaces voor ‘kantoortoepassingen’. Platformafhankelijkheid blijft een punt van zorg. Terwijl in de tools het verschil tussen browseromgeving en Windows-platform nog prominent aanwezig is, dienen zich nieuwe uitdagingen al weer aan (WAP, 3D-visualisatie, etc.). Het duurt lang voordat hulpmiddelen nieuwe ontwikkelingen op het gebied van gebruikersinterfaces ondersteunen, waardoor de benodigde inspanning voor het maken van state-of-the-art interfaces nog steeds groot is en zelfs groter wordt.

Drs. Rob Maijers

is adviseur bij het SERC te Utrecht en is gespecialiseerd in software-architecturen, intranet/Internet-technologie en human computer interaction.