Smart Visual Interfaces – adaptive Anzeige graphischer Modelldaten

Dipl.-Inf. Georg Fuchs
Prof. Dr.-Ing. habil. Heidrun Schumann
Universität Rostock, Institut für Informatik

Die wachsende Größe und Komplexität von 3D-Modellen im CAx-Umfeld erfordert neue Lösungsansätze für die Erzeugung und Verarbeitung der Modelle. Das Netzwerk GO-3D untersucht, wie durch moderne Computergrafik die Kommunikation mit den Modellen durch Präsentation, Interaktion und Informationsbereitstellung verbessert werden kann. Ein neuer Ansatz ist das Konzept der „Smart Visual Interfaces". Das Prinzip besteht darin, dem Anwender über eine adaptive Benutzerschnittstelle genau die graphischen Daten und in genau dem Detaillierungsgrad anzuzeigen, die er für seine aktuelle Aufgabe benötigt. Es wird also entschieden *für wen* (Anwender) *was* (welche Informationen) *warum* (für welche Aufgabe) und *wo* (auf welchem Ausgabegerät) angezeigt werden soll. Entsprechend der Beantwortung dieser Fragen wird die Modelldarstellung erzeugt, und sobald sich der Kontext ändert, passt sich die Darstellung adaptiv an.

Die Grundlage unserer Entwicklungen eines „Smart Visual Interface" bildet der Anwendungshintergrund eines mobilen Wartungsszenarios. Der Instandhaltungsingenieur soll in seinen Arbeitsabläufen unterstützt werden, und dazu müssen die Modelle und Informationen der Anlagen sowie notwendige Arbeitsschritte und Anfahrtsskizzen auf einem PDA ausgegeben werden. Dies stellt eine besondere Herausforderung dar, da die Ressourcen und insbesondere die Displayfläche mobiler Endgeräte beschränkt sind. Unser Schwerpunkt liegt auf der Anzeige graphischer Daten. Dabei betrachten wir drei Datenklassen:

- Rasterdaten (z.B. gescannte Abbildungen aus Bedienungsanleitungen)
- Vektordaten (Schemazeichnungen, Schaltpläne, …)
- 3D-Modelle in Form von Dreiecksnetzen (Anlagen, Maschinenbauteile, …).

Das *Was*, *Wo* und *für Wen* ist damit festgelegt. Die Adaption der Präsentation erfolgt anhand der Arbeitsschritte, wird also durch das *Warum* gesteuert. Hierfür brauchen wir ein Modell, das die Arbeitsschritte sowie die notwendigen Informationen für die Anpassung der visuellen Ausgabe bereitstellt. Im Bereich Nutzerinterface-Design wird mit Taskmodellen gearbeitet. Diese beinhalten eine hierarchische Struktur von Aufgaben und Teilaufgaben sowie die Spezifikation von Constraints. Ein weit verbreitetes Taskmodell ist der ConcourTaskTree (CTT, vgl. [PMM97]). Jeder Knoten des Baums repräsentiert eine Aufgabe, die Kindknoten repräsentieren die zugehörigen Teilaufgaben, die durch orientierte Kanten temporal und kausal in Beziehung gesetzt werden. Die Blätter des Baumes beschreiben Aktionen, die für elementare Aufgaben stehen. Für jede Aktion wird spezifiziert, ob sie vom Nutzer, vom System oder durch Interaktion von Nutzer und System ausgeführt wird. Ein Pfad durch das Taskmodell beschreibt eine konkrete Sequenz von Aufgaben, die zur Lösung eines spezifischen Problems abgearbeitet werden muss. Taskmodelle werden manuell erstellt, so auch in unserem Beispiel, dem Wartungsszenario. Dazu werden typische Arbeitsabläufe zur Wartung von Anlagen in CTT-Notation erfasst. Die automatische Anpassung üblicher WIMP-Userinterfaces (**W**indows, **I**cons, **M**enus, **P**ointer) auf Basis der CTT-Notation ist ohne weiteres möglich. (vgl. z.B.[PS02])

Erweiterung der Aufgabenbeschreibung

Für die Anpassung der graphischen Ausgabe muss das Taskmodell erweitert werden. Unser Ansatz ist ein PAGE/FEATURE – Konzept, das die folgenden drei Schritte umsetzt:

1. Aufbereitung der graphischen Daten und Ablegen der Informationen in PAGEs. Zuordnung einer PAGE zu einem oder mehreren Knoten des Taskmodells;
2. Definition von FEATUREs einer PAGE zur Beschreibung der für eine Aufgabe relevanten Aspekte;
3. Traversieren des Taskmodells und Adaption der graphischen Ausgabe auf Basis der PAGE/ FEATURE-Angaben.

Die Definition von PAGEs und FEATUREs sollte genau wie die Definition des eigentlichen Taskmodells manuell vorgenommen werden. Dazu wurde für jeden Datentyp (Rasterdaten, Vektordaten, 3D-Netze) ein einfach zu bedienendes Autorenwerkzeug entwickelt.

Autorenwerkzeuge für die grafische Datenaufbereitung

Das Autorenwerkzeug für *Rasterdaten* (vgl. Abb. 1) stellt als PAGE das entsprechende Pixelbild bereit, zeigt dies in einem Fenster an und erlaubt eine manuelle Segmentierung. Durch das Nachzeichnen von Konturkanten oder die Spezifikation von Bildbereichen werden Segmente als FEATUREs definiert. Die PAGE wird einem oder mehreren Knoten des Taskmodells zugewiesen. Die FEATUREs werden mit normierten Relevanzwerten assoziiert, die ihre Bedeutung für die zu lösenden Teilaufgabe ausdrücken. Außerdem können weitere Informationen an ein FEATURE gebunden werden, wie z.B. eine Farbangabe zur Akzentuierung von Bildteilen oder kurze Texte für die Beschriftung.

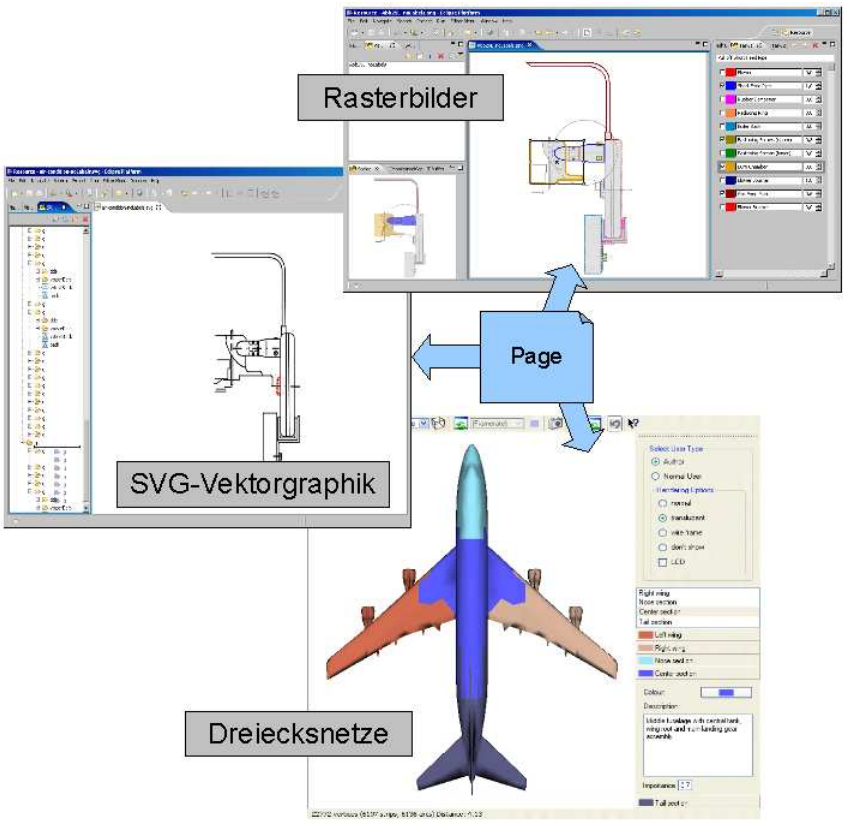

Abbildung 1 Autorenwerkzeuge für die Erstellung von PAGEs und die Definition von FEATUREs für die unterschiedlichen Datentypen.

Das Autorenwerkzeug für *Vektordaten* beschreibt eine PAGE im SVG-Format und nutzt zwei wichtige Eigenschaften von SVG aus: Gruppierung und Symboldefinition. Die Gruppierungsfunktionalität wird angewendet, um graphische Elemente wie Linien und Polygone zusammenzufassend als FEATURE zu definieren und für bestimmte Aufgaben zu priorisieren. Dabei ist darauf zu achten, dass durch die so vorgenommene Gruppierung bei der Darstellung und Traversierung der Struktur in „back-to-front"-Ordnung keine Fehler entstehen. Deshalb wird ein kontinuierliches visuelles Feedback gegeben, so dass der Nutzer gegebenenfalls vorgenommene Gruppierungen korrigieren kann. Symbole sind spezielle Gruppierungen von häufig vorkommenden Teilen der Vektorgrafik. Die Symboldefinition wird dahingehend angewendet, dass je FEATURE mehrere Symbolgruppen mit jeweils unterschiedlichen Genauigkeitsstufen spezifiziert werden, von denen eine anhand des Relevanzwertes ausgewählt und angezeigt wird. Die PAGE wird einem oder mehreren Knoten des Taskmodells zugeordnet, die normierten Relevanzwerte werden pro Aufgabe zugeordnet und weitere Informationen für den Darstellungsprozess abgelegt, wie z.B. Modifikationen von Attributwerten zur Akzentuierung/ Deakzentuierung, Transformationen zur Erzeugung von Explosionszeichnungen oder Texte für die Beschriftung.

Das Autorenwerkzeug für *3D-Netze* beschreibt eine PAGE als segmentiertes Dreiecksnetz. Für die FEATURE -Definition werden folgende drei Schritte abgearbeitet:

1. Segmentierung
2. Modell Inspektion
3. Segmentpriorisierung

Falls das Dreiecksnetz bereits aus semantisch sinnvollen Teilnetzen besteht, kann der Segmentierungsschritt entfallen. Anderenfalls wird das Dreiecksnetz automatisch segmentiert. Dazu wird das Netz vereinfacht und eine Level-of-Detail-Hierarchie erzeugt. Dadurch lässt sich die Performance verbessern und es liegen zudem verschiedene Genauigkeitsstufen für die spätere Adaption der Darstellung vor. Nach dieser initialen Segmentierung wird das Modell in Falschfarben dargestellt, um die einzelnen Segmente zu unterscheiden. Im zweiten Schritt, der Modell-Inspektion, verifiziert der Autor das Ergebnis des Segmentierungsprozesses für eine Genauigkeitsstufe des Modells. Dafür werden die üblichen Funktionen zur Rotation des Modells, zum Zooming oder zur Anpassung der Viewing- Bedingungen angeboten. Das Segmentierungsergebnis wird bei der Inspektion so korrigiert, dass es den Erfordernissen der zu lösenden Aufgaben entspricht. Dazu können Segmente durch das Hinzufügen oder Entfernen von Dreiecksstrips vergrößert bzw. verkleinert werden. Außerdem können Segmente zusammengefasst bzw. ein einzelnes Segment für eine weitere automatische Unterteilung ausgewählt werden. Mit einer 3D-Lupe lassen sich außerdem Bereiche von Interesse definieren. Die Dreiecke innerhalb so eines Bereiches werden als Segment zusammengefasst. Nach dem Inspektionsschritt ist das Dreiecksnetz in semantisch sinnvolle Segmente unterteilt. Diese Unterteilung wird auf alle Genauigkeitsstufen des Netzes übertragen und als Menge der FEATUREs definiert. Die PAGE wird an einen oder mehrere Knoten des Taskmodells gebunden und die FEATUREs werden mit normierten Relevanzwerten assoziiert. Daneben können im Taskmodell weitere Informationen abgelegt werden, dazu gehört z.B. die Position der Kamera, damit garantiert ist, dass die für eine Aufgabe wichtigen Objektteile auch sichtbar sind. Außerdem können Angaben zur Modifikation von Attributwerten und Renderingstilen sowie Texte für eine Beschriftung abgelegt werden.

Automatische Adaption der graphischen Präsentation

Im Ergebnis der manuellen Datenaufbereitung liegt ein angereichertes Taskmodell vor, das alle Angaben für eine adaptive Präsentation der graphischen Daten enthält. Die automatische Adaption läuft in 3 Schritten ab:

1. Adaption der Geometrie
2. Adaption der Darstellung
3. Annotation

Zunächst erfolgt im ersten Schritt die *automatische Adaption der Geometrie*. Dazu gehören zwei Aspekte: Aufteilung der Ausgabefläche und Level-of-Detail- Auswahl. Die Ausgabefläche wird automatisch entsprechend der Relevanzwerte im Taskmodell so aufgeteilt, dass für wichtige FEATURE viel Platz und für weniger interessante FEATURE auch weniger Platz zur Verfügung steht. Dazu wurde ein Belt-based Focus & Context-Ansatz entwickelt (vgl. [FRSF06]). Je nachdem, in welchem Belt ein FEATURE dargestellt wird, erfolgt eine Vergrößerung (oder 1-1-Abbildung) bzw. eine Verkleinerung der entsprechenden Objekte. Dieser Ansatz ist für alle drei Datentypen anwendbar. Bei Rasterdaten werden Pixelbereiche skaliert, bei Vektorgrafik SVG-Elemente und bei Dreiecksnetzen Segmente. Die automatische Level-of-Detail-Auswahl wird nur für SVG und Dreiecksnetze durchgeführt. Entsprechend der Relevanzwerte der einzelnen FEATURE werden wichtige FEATURE in einer hohen Genauigkeit angezeigt, weniger relevante FEATURE in einer geringern Auflösung. Abbildung 3 veranschaulicht die geometrische Adaption.

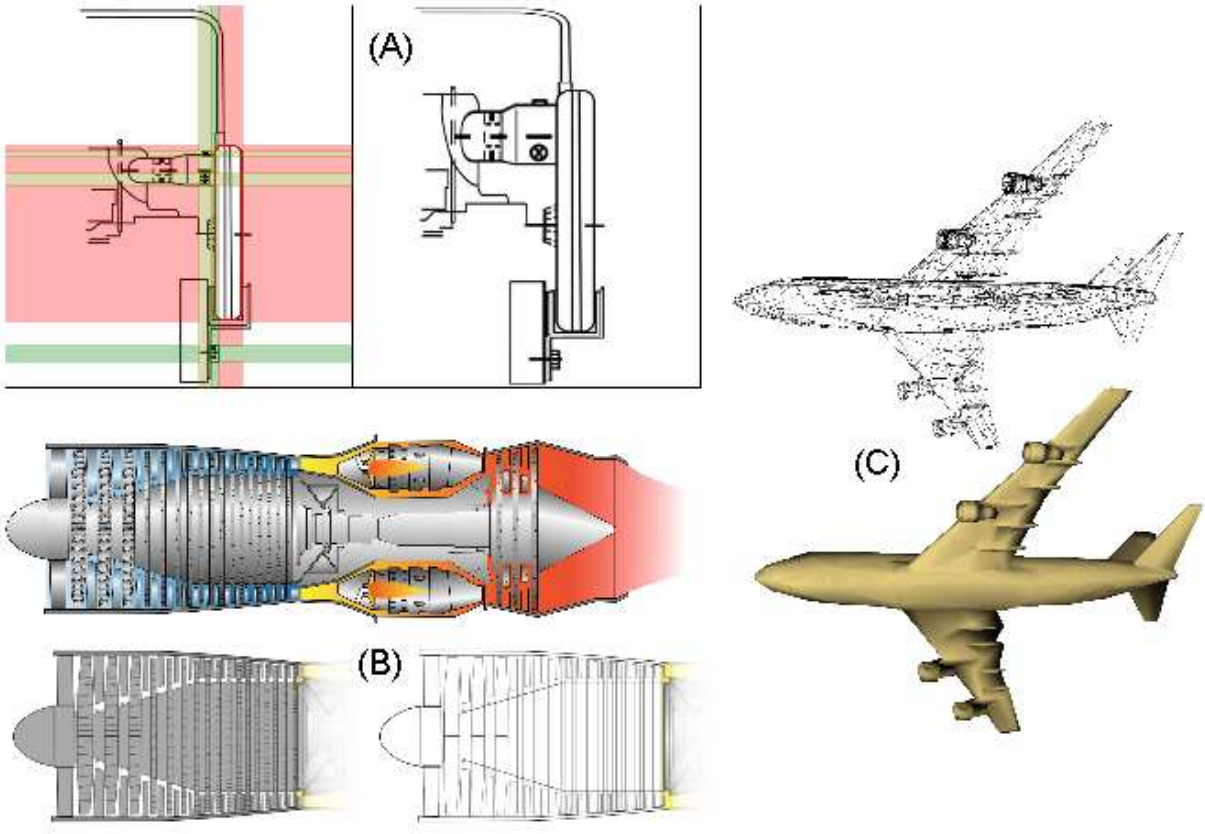

Abbildung 2 Beispiele für die geometrische Adaption: Belts für drei FEATUREs und resultierende Vergrößerung wichtiger Bildbereiche (A), LoD-Auswahl in SVG (B) und bei Dreiecksnetzen (C).

Im zweiten Schritt erfolgt die *automatische Adaption der Darstellung*. Hierbei werden die Relevanzwerte ausgewertet, um Farb- und Transparenzwerte für eine akzentuierte Darstellung

zu setzen. Zusätzlich werden die an ein FEATURE gebundenen Angaben zur Modifikation von Attributwerten und Renderingstilen ausgewertet. Das bedeutet z.B., dass SVG- Elemente transformiert werden, um durch Explosion wichtige Objektteile frei zu stellen, oder dass bei Dreiecksnetzen der Blickpunkt gesetzt oder weniger wichtige Segmente als Wireframe dargestellt werden. Abbildung 4 zeigt Beispiele für die Adaption der Darstellung.

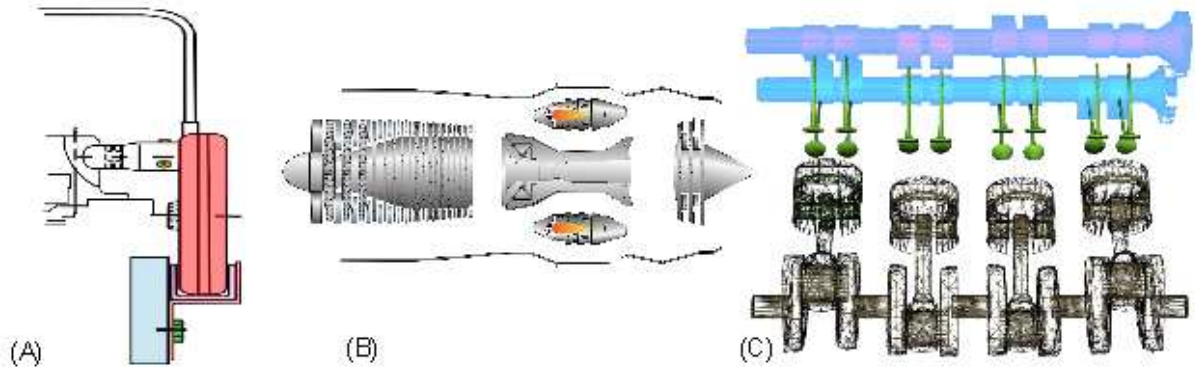

Abbildung 3 Beispiele für die Adaption der Darstellung: (A) Anpassung von Farb- und Transparenzwerten , (B) Freistellen wichtiger Elemente und (C) Wahl unterschiedlicher Renderstile.

Annotation der Präsentation

Zum Schluss wird die *Darstellung annotiert*. Anhand der Relevanzwerte wird automatisch entschieden, für welche FEATURE und in welcher Größe und Umfang die assoziierten Texte angezeigt werden. Eine adäquate Beschriftung ist ein komplexes Problem. Durch die aufgaben-bezogene Aufteilung der Ausgabefläche steht für die Darstellung und Annotation wichtiger Objektteile aber auch ein größerer Platz zur Verfügung. Die Annotation erfolgt für alle Datentypen auf die gleich. Zunächst wird ein id-Buffer der aktuellen Ansicht erzeugt (vgl. [FLH+06]). Der id-Buffer speichert jedes FEATURE in einer separaten Farbe als Pixelrepräsentation. Unbelegte Pixel markieren den Hintergrund, der für die Beschriftung genutzt werden kann. Den Hintergrundpixeln werden Distanzwerte zu den Objektgrenzen zugewiesen und in einem so bezeichneten Distanzfeld abgelegt. Damit wird garantiert, dass die Label so nahe wie möglich an den Objekten platziert werden können. Die Annotation erfolgt nun Schritt für Schritt. Zunächst wird das FEATURE mit dem höchsten Relevanzwert beschriftet, das Distanzfeld aktualisiert und dann mit dem FEATURE mit dem nächst höheren Relevanzwert fortgefahren. Dieser Prozess wird solange fortgesetzt, bis nicht mehr ausreichend Platz für Annotationen zur Verfügung steht. In der Regel können so nicht alle Texte einer PAGE angezeigt werden. Deshalb ist eine Beschriftungslupe integriert. Dort, wo die Lupe platziert ist, werden zusätzlich Annotationen nach Bedarf eingeblendet. Es sei noch angemerkt, dass unsere Implementation mit einer Audioausgabe gekoppelt ist, so dass längere Texte automatisch vorgelesen werden. Abbildung 5 zeigt Beispiele mit Annotationen.

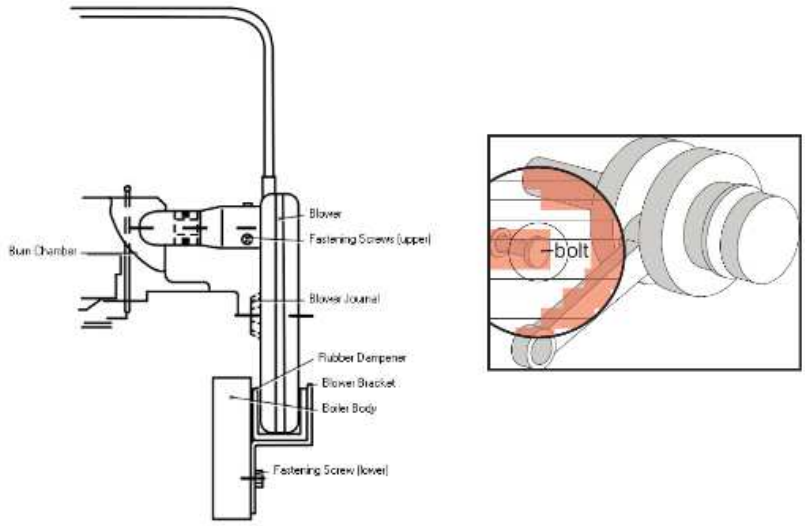

Abbildung 4 Dynamische Annotation der Darstellung und interaktive Beschriftungslinse

*Die automatische Anpassung der visuellen Modellrepräsentation garantiert, dass die zur Bearbeitung einer Aufgabe wichtigen Objektteile in der graphischen Präsentation sichtbar und akzentuiert sind. Das ist besonders im Umfeld eines mobilen Wartungsszenarios eine wichtige Eigenschaft, da auf Grund der beschränkten Ausgabefläche nicht alle Objekte in voller Genauigkeit dargestellt werden können. Trotzdem muss es dem Anwender erlaubt sein, diese initiale Ansicht jederzeit zu verändern. Deshalb stehen Funktionen zur Verfügung, die es erlauben, die automatisch gesetzten Parameter interaktiv zu verändern und damit andere Sichten zu erzeugen. Weiterführende Informationen zu einzelnen Aspekten des vorgestellten Ansatzes sind zu finden in [*FRSF06*] (mobiles Wartungssoftware), [*FLH+06*] (Annotation), [*FHS08*] (adaptive 3D-Modelldarstellung).*

[FHS08] G. Fuchs, M. Holst, H. Schumann. 3D Mesh Exploration for Smart Visual Interfaces. *Proc. 10th Intl. Conference on Visual Information Systems*, Springer LNCS 5188, S. 80–91, 2008.

[FLH+06] G. Fuchs, M. Luboschik, K. Hartmann, K. Ali, T. Strothotte, H. Schumann. Adaptive labeling for interactive mobile information systems. *Proc. IV'06*, S. 453–459, 2006.

[FRSF06] G. Fuchs, D. Reichart, H. Schumann, P. Forbrig. Maintenance Support – Case Study for a Multimodal Mobile User Interface. *Multimedia on Mobile Devices II, Proc. SPIE'06*, Vol. 6074, 2006.

[PMM97] F. Paterno, C. Mancini, and S. Meniconi. ConcurTaskTrees: A diagrammatic notation for specifying task models. *Proc. Interact'97*, S. 362–369, 1997.

[PS02] F. Paterno and C. Santoro. One model, many interfaces. *Proc. 4th International Conference on Computer-Aided Design of User Interfaces*, S. 143–154, 2002.