Generating Adaptable Multimedia Software from Dynamic Object-Oriented Models: The OBJECTWAND Design Environment

> Prof. Dr.-Ing. Christian Märtin Dipl.-Inf. (FH) Michael Humpl

Fachbereich Informatik Fachhochschule Augsburg D-86161 Augsburg (Germany)



Introduction

- From AME to OBJECTWAND
- Automated, model-based design of adaptable multimedia applications
- Designer-system cooperation over the whole life cycle
- Looking for automation tasks in life cycle activities
- Integrating multimedia-specific design requirements into OO application models
- Future directions



From AME to OBJECTWAND

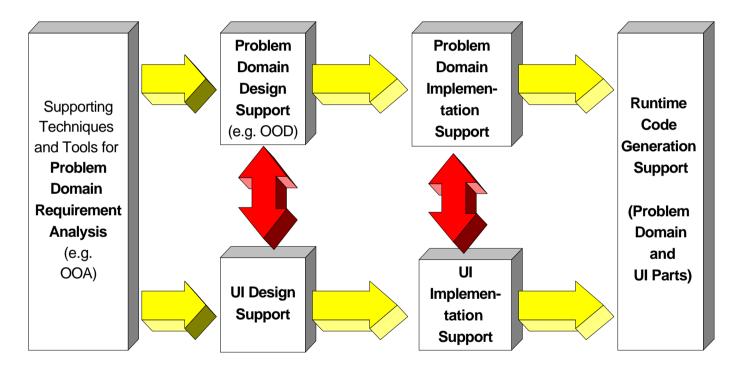
• **OBJECTWAND**

- evolutionary successor to AME, a design and generation environment for interactive applications based on dynamic OOA/D-models
- Related work on model-based systems
 - current state-of-the-art is documented in [Vanderdonckt, 1996]
- OBJECTWAND introduces
 - improved features for detailed OO modeling and design
 - easy integration of standard multimedia elements
 - better cooperation between interactive and automated design and generation tools
 - a new application generator component (Borland Delphi)



Integration of the life cycles

OO technology as the common denominator



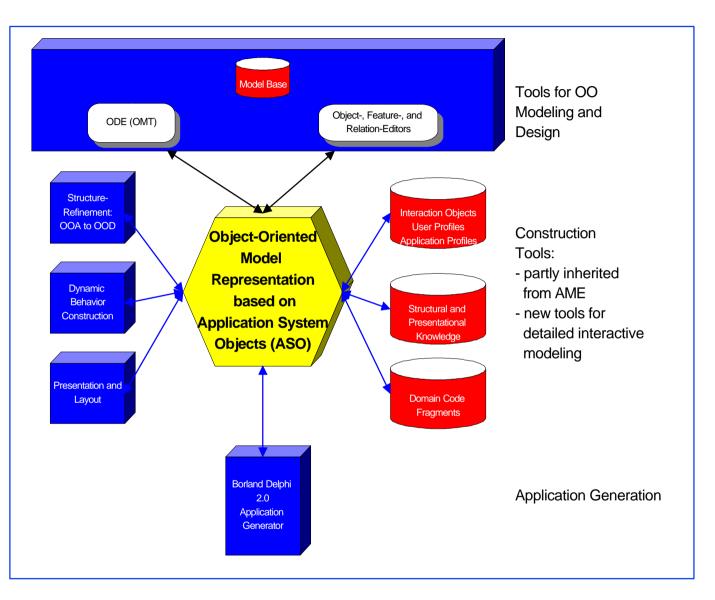
Unifying specification and generation

- Balancing the ratio of flexibility vs. automation
 - The designer's choice at each life cycle activity
 - accept the intermediate model as proposed by the system
 - partly modify or extend the intermediate model interactively
 - integrate reusable components into the intermediate model
 - Model refinement knowledge
 - standard knowledge for standard design problems
 - optional domain or user specific knowledge for adaptation
 - experience of the designer
 - technical knowledge for automated system integration



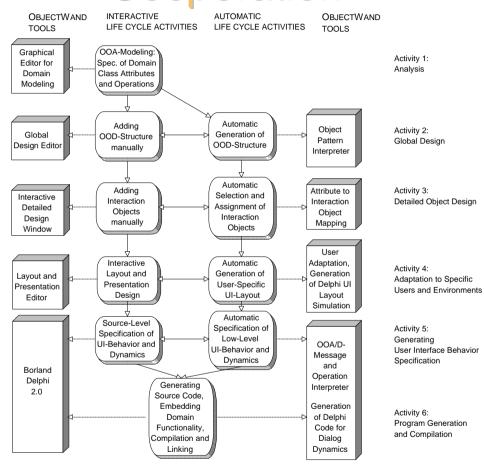
OBJECTWAND Architecture

- Three tool categories:
 - OO modeling/ design
 - application construction
 - generation
- ASO representation scheme
- Target applications:
 - Business domain





Life Cycle Support and Designer-System Cooperation



FH Augsburg - FB Informatik

--HCI International '97 - San Francisco

OOA Modeling

- Example OOA model specification
 - attributes
 - operations
 - gen/spec relations
 - aggregations
 - associations
 - message channels

			C	bjekt - [CADI	MAEF	R1.0MN]				•
-	<u>D</u> atei	<u>B</u> earbeiten	<u>E</u> lement	<u>C</u> odeerzeu	gung	<u>A</u> nsicht	<u>F</u> enster	<u>O</u> ptionen	<u>H</u> ilfe	\$
	F B	<u>ə xec</u>		<u>→ </u>	<u> </u>		<u></u>		· · · I	
		· · · · · · · · · · · · · · · · · · ·								1
				Clas	s 1					
	Attrib Attrib	oute_1 : Attrib oute_2 : Qual oute_3 : Qual p, Attr_41, At	lif_Attribute lif_Attribute	e_Type_1 e_Type_2	oup, t	ype_41, t	ype_42, <u>c</u>	jroup_end		
••••	Oper	ration_1(Para ration_2 : "Ac ration_3 : NIL	ctivate:Cla):TF	lue				
		ass 2	Cla	ss 3		Class 4		Class 5		
::							·····			4
	 Ir Hilfe di	rücken Sie E1							Seite: .	<u> </u> → 1



Generating the window hierarchy

- Mapping of complex OOA classes to windows, if possible
 - expanding gen/spec relations
 - exploiting attributes, attribute types, grouped attributes
 - exploiting aggregational object patterns
- Creating the menu- and command hierarchy
 - exploit target environmental constraints and synonym lists
 - assignment of OOA operations to menu entries or buttons
- Refine generated results interactively



Assigning interaction objects

- Exploiting qualified attribute types, if available:

- map the attribute type to a specific OOD object pattern
- assign abstract interaction objects (AIOs) to the OOD objects
- If not: activate a number of rule groups
 - exploit content oriented meta data
 - exploit synonym lists to find the correct qualified data type
 - exploit the cardinality of attributes and/or operations
 - refinement of already assigned AIOs
 - rules in user and environment profile
- Alternatively
 - assign or modify the AIO manually



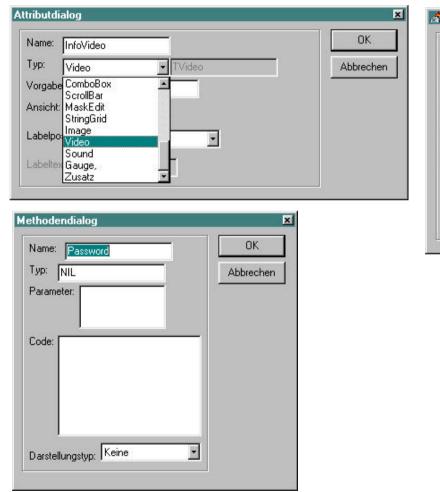
UI behavior spec

- **OBJECTWAND** exploits

- OOA/D-messages, relational OOA/D patterns (including OOA/D class features),
- detailed-design-level object groups for target functionality (e.g. non-standard multimedia behavior)
- to automatically generate
 - menu/command activation methods, inter-object communication by message-passing, domain method code integration
- Designer adds domain-specific functionality and dynamics
 - OBJECTWAND provides method- and message-editors



OBJECTWAND Detailed Design Environment





FH Augsburg - FB Informatik --

OBJECTWAND Detailed Design Environment

K: Textum Internet Name: ac_component4 Copion: K: Decide internet Name: ac_component4 Copion: K: Decide internet Name: ac_component4 Decide internet Name: ac_component4 AggeogatorFiom Transmort: Decide internet Name: ac_component4 AggeogatorFiom Name: OK:	Image: Second	Standard Zusatzlich Win95 Datenzugnit Datenzugnit <th>atensteuerung Win 3.1 Dialoge System QRepu</th> <th>nt UCX Beispiele </th> <th></th>	atensteuerung Win 3.1 Dialoge System QRepu	nt UCX Beispiele 	
Vinit Unit2 Unit3 TPasswordIn = class(TForm) LabelOfP_Eingabe: TLabel; P_Eingabe: TLatel; OK: TButton; procedure OKClick(Sender: TObject); procedure AbbruchClick(Sender: TObject); procedure GetTheMessage(var Msg:TMessage); mess end; 	Eigenschaften Ereignisse Cancel False ▲ Caption OK Cursor orDefault Default False DragCursor orDrag DragMode dmManual Enabled True +Font (TFont) Height 25 HelpContext 0 Hint Left 151 ModalResult mtNone	Bitte Password eingeben:	Datei Work Benutzerprofil Info Passworddialog Attibute HasPassword : Boolean NeuesPassword : String BestPassword : String Operationen OK:NIL Abbruch:NIL	Interner Name: as_component4 AggregationFrom Tagebuch: 1:1 Assoziation NachrichtTo	
	Unit3.pas Unit3.pas Unit1 Unit2 Unit3 TPasswordIn = class(LabelOfP_Eingabe: TL P_Eingabe: TEdit; OK: TButton; Abbruch: TButton; procedure OKClick(Se procedure AbbruchCli private public procedure GetTheMess end;	TForm) abel; nder: TObject); ck(Sender: TObject); age(var M sg:TMessage); mess		NachrichtFrom	



Additional Features

- Generation of standard UI layouts (inherited from AME)

- choosing dialogbox and window layout types automatically
 - based on types and number of certain AIOs within a window
 - based on inter-object-relations (patterns) among OOA/D classes
- adaptable standard layout alternatives,
- individual rules in user/env. profiles
- Interactive modification of generated layout and interaction object features (in Delphi)
- Adding domain- or database-specific program components, before creating runtime applications (in Delphi)

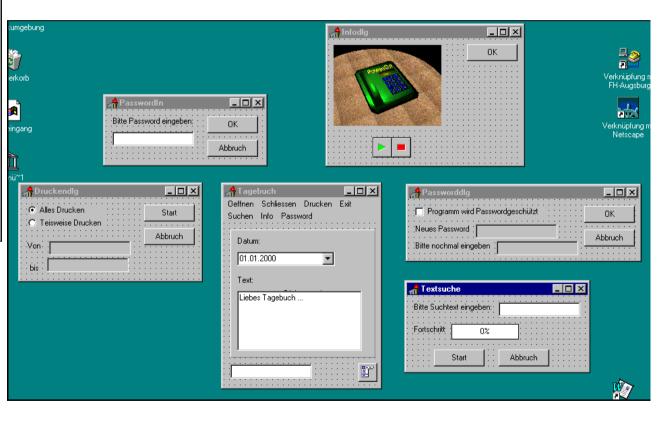


Target application generation

- Scanning/Parsing the detailed specification
- Generating C++-code, callbacks, defaults
- Integration of existing domain method code
- Integration of standard dialog boxes and behavior
- Integration of reusable major application parts
- Activating the Borland C++-compiler
- As an alternative a UIMS interface exists

Generated Target Application with its Original OOA Model

Infodig	Q		Text	Textsuche	
InfoVideo : Tvideo = IN	F0.AVI		Eingabe: St Fortschritt: T		
ок]	Stert		
Passwordin	1	ŶŶ			
P_Eingabe :String OK Abbruch		Tagebuch		Password	
		Deteineme : String Password Str : String	¢-ر_		
	- Lo	Oeffnen Schliessen Drucken Exit Textsuche Info	•	OK Abbruch	
		Pessword			





FH Augsburg - FB Informatik

--

HCI International '97 - San Francisco

Conclusions and future work

- Benefits

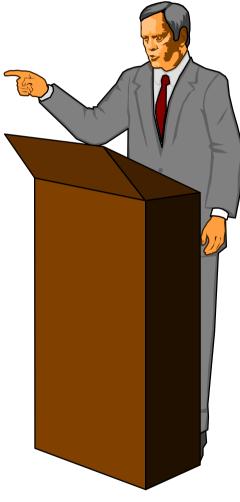
- complete prototypical life cycle support for computer-aided development of interactive systems
- automating many standard design tasks for modeling interactive systems

– Current and future work

- For improving the degree and the quality of automation, we will
 - look for exploitable software patterns at all life cycle activities
 - define standard model fragments at all life cycle activities
 - extend the expressional power of our representation scheme (ASO)
 - look for alternative ways for modeling system level dialog dynamics (e.g. including dynamic patterns and task modeling)
 - look for synergies with other disciplines and similar problems



Thank you for your attention!





-- HCI International '97 - San Francisco