Ergebnisse einer Evaluierung von CASE-Tools

Drum prüfe ...

Michael Kunz, Jan Geißelmann, Andreas Hierholzer, Georg Herzwurm

17 Werkzeuge im Test: Sind konventionelle CASE-Tools altmodisch und objektorientierte modern? Welche Kriterien spielen bei der Auswahl eine Rolle? Mit diesen Themen beschäftigt sich eine Studie der Universität Köln, die eine Orientierungshilfe im Dschungel objektorientierter und konventioneller CASE-Werkzeuge geben will.

Sie machen jetzt noch eine CASE-Studie?' – So oder ähnlich lautete eine Reihe von Reaktionen, als sich der Lehrstuhl für Wirtschaftsinformatik, Systementwicklung, der Universität Köln Anfang 1993 entschloß, eine Studie zum Markt des Computer Aided Software Engineering (CASE) in Deutschland anzufertigen. Diese Skepsis resultierte zum einen aus dem Ende der Euphorie, die offenbar unvermeidlich bei jeder neuen Technologie im Bereich der Softwareentwicklung zuerst aufkommt und dann abflaut, zum anderen aus der Existenz früherer Veröffentlichungen zu diesem Thema, zum Beispiel von Ploenzke, PSI oder Ovum.

Über eine Marktübersicht hinaus sollte die neue Studie eine Orientierungshilfe für Auswahl und Einführung konventioneller und objektorientierter CASE-Werkzeuge bereitstellen, indem sie den 'State of the Art' dieser Technologie in Deutschland auf Anbieter- und Nachfragerseite untersucht. Die Perspektive der Anwender wurde durch eine empirische Untersuchung berücksichtigt.

Wie alles anfing ...

Informationen über die Hersteller gewann das Bearbeitungsteam in drei Etappen: Neben einem Fragebogen, den es an 60 Hersteller verschickte, entwickelten die Mitarbeiter am Lehrstuhl in einer zweiten Stufe einen Katalog mit über 300 Kriterien, bei dem vorwiegend Methodenaspekte und weniger releaseabhängige Gesichtspunkte im Vordergrund standen. Diesen Kriterienkatalog, den ein Diplomand auch als Software realisierte, füllten 28 Anbieter aus.

Die letzte Stufe bildeten Testinstallationen von 17 marktbedeutenden Tools, mit denen ein bis zu elfköpfiges Evaluierungsteam bis Januar 1994 eine Fallstudie aus dem kommerziellen Bereich bearbeitete. Zur grundsätzlichen Einschätzung von konventionellen und objektorientierten CASE-Tools standen fünf Anforderungen im Vordergrund:

- Methodenorientierung
- durchgängige Methodenunterstützung
- Teamfähigkeit durch Repository
- Benutzerfreundlichkeit
- Anpaßbarkeit und Erweiterbarkeit.

Basis von CASE-Tools sind Methoden, also Vorschriften, mit denen der Systementwickler planmäßig Arbeitsergebnisse erzielen kann. Grafische

Beschreibungsmittel wie Daten- oder Klassenmodelle unterstützen diesen Prozeß. Neben konventionelle strukturierte Methoden sind eine ganze Reihe objektorientierter Ansätze getreten. Für beide Richtungen existieren jeweils eigene CASE-Werkzeuge, wobei der Markt der objektorientierten Varianten derzeit modisch im Trend liegt und daher sehr dynamisch ist.

CASE-Tools stellen Editoren für die Beschreibungsmittel des jeweiligen Ansatzes zur Verfügung. Eine durchgängige Methodenunterstützung zeichnet sich dadurch aus, daß sie die Ergebnisse in nach- und eventuell auch vorgelagerte Phasen der Entwicklung übernimmt oder transformiert, ohne daß Informationen verlorengehen.

Voraussetzung für jeden ernsthaften Einsatz einer Entwicklungsumgebung ist ein Repository, in dem alle Projektdaten gespeichert und auf die alle beteiligten Mitarbeiter zugreifen können. Erweiterbare Werkzeuge ermöglichen es, ihre Grundfunktionalität um firmenspezifisch wichtige Funktionen zu ergänzen.

Strukturierte Tools sind ausgereift

An diesen grundlegenden Anforderungen hat das Evaluierungsteam konventionelle und objektorientierte Werkzeuge gemessen. Das erste CASE-Tool für die strukturierten Methoden erschien bereits zu Beginn der 80er Jahre. Daher sind die Produkte mit diesem Ansatz tendenziell ausgereifter als ihre objektorientierten Gegenstücke.

Ein Repository sorgt bei den konventionellen Produkten bei wenigen Ausnahmen – hier ist das Tool Natural Engineering Workbench (NEW) zu nennen – für Teamfähigkeit. Als Grundlage dienen dafür häufig die relationalen Datenbanksysteme. Werkzeuge wie Maestro II von Softlab oder Teamwork von Cadre setzen dagegen weiterhin auf ihre eigenen Datenhaltungssysteme. Als Grund hierfür führen die Hersteller eine bessere Performance bei größeren Entwicklergruppen an.

Die PC-basierten Programme sind benutzerfreundlicher als die meisten Unix-Produkte. Genau umgekehrt verhält es sich, wenn es um Erweiterungsmöglichkeiten und Anpaßbarkeit geht: So bieten Unix-Tools wie Teamwork Testwerkzeuge und Erweiterungen an, mit denen Entwickler beispielsweise textuelle Anforderungen definieren und verfolgen können.

Für jeden das richtige

Im PC-Bereich gibt es für fast jedes Einsatzgebiet ein passendes konventionelles Produkt: Die Auswahl reicht von einfachen Lösungen für einige hundert Mark pro Arbeitsplatz bis zu High-End-Lösungen im (Novell-)Netz auf Basis relationaler Datenbanken mit ausgefeilter Funktionalität für 20 bis 50 Entwickler. Der Preis pro Arbeitsplatz für solche Tools liegt im fünfstelligen Bereich.

Die Anzahl der Methoden, die den objektorientierten Vertretern zugrundeliegen, übersteigt sicherlich ein Dutzend. Im Kern verwenden sie ganz ähnliche Beschreibungsmittel, wenn man von Unterschieden im Aussehen, in der Symbolik und in Details der Vorgehensweisen abstrahiert. Dementsprechend unterstützen alle evaluierten Tools das Erstellen von Klassen- und Zustandsübergangsdiagrammen. Nur SOM-CASE verfolgt einen völlig anderen methodischen Ansatz.

Aus dieser Vielzahl von Methoden konnte sich bisher keine als Standard durchsetzen. Allerdings besteht eine gewisse Dominanz der Object Modelling Technique (OMT) nach Rumbaugh et. al., ein Ansatz, der bei etwa zwei Drittel der untersuchten Werkzeuge zum Einsatz kommt.

Zwischen PC- und Unix-Werkzeugen differiert die Qualität der Methodenunterstützung stark. Am unteren Ende stehen PC-Lösungen wie ObjecTool und OMTool, deren Funktionalität sich im wesentlichen auf das Entwerfen von Diagrammen beschränkt. Eine gute methodische Unterstützung bieten die leistungsfähigen Unix-Produkte wie Westmount OMT, StP/OMT und Teamwork OOA/OOD. Sie sorgen

iX-TRACT

- Ein Team an der Universität Köln evaluierte 17 CASE-Tools anhand von 300 Kriterien.
- Tendenziell sind konventionelle CASE-Tools ausgereifter als ihre objektorientierten Gegenstücke.
- Die Leistungsfähigkeit objektorientierter Unix-CASE-Tools übersteigt die der PC-Lösungen.
- Es ist im Moment mit keinem Tool möglich, Programme zu entwickeln, die leistungsfähigere und ohne grundsätzliches Redesign der Anwendungen änderbare Verteilungsarchitekturen (vergleiche CORBA der OMG) einsetzen.

EVALUIERTE CASE-TOOLS IM ÜBERBLICK

Produkt	Hersteller	Betriebssysteme					grafische Beschreibungsmittel
		DOS	MS Windows	OS/2	Unix	Sonstige	
ADW	Knowledgeware	●		●	●		ERD, AD, DFD, SC, M-Spek
case4/0	microTool		●		●		ERD, JD
DDB-Conceptor/DDB-CASE	Delta		●	●	●		ERD, DFD, FB, M-Spek
Excelerator	Intersolv	●		●			ERD, DFD, K-Spek
IEF	TI Information Engineering		●	●	●	MVS	AD, SC, ELH, ERD, DDD
Innovator	MID	●	●	●	●	●	DFD, SERM, SC, M-Spek, RT
Maestro II	Softlab	●		●	●		ERD, FB, Fktfld
ObjecTool	Rösch Consulting	●	●		●		OAOD
OM-Tool	IQProducts	●	●		●		OMT
NEW (PredictCASE)	Software AG		●				DFD, ERD, BFD, KFD
ProMod	CAP debis SSP		●	●	●	VMS	DFD, ERD, FB, SC, K-Spek, M-Spek
Software through Pictures	IDE				●		DFD, ERM, OOD, SC
SDW	SDW Software	●	●				DFD, ERD, SC
SOM-CASE	Uni Bamberg	●	●		●		SERM, SOM
Systems Engineer	LBMS		●				DFD, ELH, ERD, MS
Teamwork	Cadre				●		ERD, M-Spek, SC, OOSA, RT
Westmount I-CASE (Yourdon, OMT)	Westmount Technology				●	VMS	CAD, DFD, ERD, M-Spek, STD, RT

● verfügbar
AD: Action Diagram
BFD: Business Function Diagram
CAD: Klassenstrukturdiagramm
DDD: Dialog Design Diagram
DFD: Datenflußdiagramm
ELH: Entity Life History
ERD: Entity-Relationship Diagram
ERM: Entity-Relationship-Modellierung
FB: Funktionsbaum
IE: Information Engineering
Fktfld: Funktionsflußdiagramm
FND: Function Net Diagram
KFD: Kontrollflußdiagramm
KM: Konfigurationsmanagement
K-Spek: Kontroll-Spezifikation
M-Spek: Mini-Spezifikation
MS: Modulstruktur

für konsistente Diagramme und eine gute Navigation zwischen ihnen.

Solche Werkzeuge sind teamfähig durch ein Repository, für das – mit Ausnahme von Teamwork – relationale Datenbanksysteme Einsatz finden. Eine Sperrung auf Diagrammebene bieten die Unix-Lösungen: Bearbeitet ein Entwickler ein Diagramm, so haben alle anderen höchstens Leserecht. Leider fehlt bei allen Produkten ein Benutzergruppenkonzept sowie ein ausgereiftes Konfigurationsmanagement.

Im Hinblick auf die Anwenderfreundlichkeit lassen sich deutliche Unterschiede und Mängel feststellen: Zum Beispiel bietet Teamwork keine intuitive Benutzerführung. Bei anderen Werkzeugen wie OMTool und ObjecTool fehlt eine Undo-Funktion und ein Hilfesystem, oder sie haben unkomfortable Diagrammeditoren. Insgesamt ist ein deutlicher Rückstand gegenüber moderner Standardsoftware erkennbar.

Durchgängige Entwicklung mit Brüchen

Teamwork hebt die Durchgängigkeit objektorientierter Methoden dadurch auf, daß für Analyse und Design zwei unterschiedliche Werkzeuge zur Verfügung stehen, zwischen denen ein Datenaustausch nicht ohne weiteres möglich ist. Alle anderen CASE-Tools erlauben in den frühen Phasen aufgrund des Einsatzes objektorientierter Methoden eine durchgängige Systementwicklung.

Problematisch erweist sich allerdings der Übergang zur Implementierungsphase. Bei keinem der getesteten Werkzeuge kann der Entwickler die Konsistenz zwischen der Kodierung der einzelnen Klassen und den Ergebnissen der frühen Phasen sichern oder überprüfen. Zur Zeit klafft eine Lücke zwischen den frühen Phasen und der Implementierung. Im Gegensatz zum Methodenbruch bei der Systementwicklung mit strukturierten Methoden ist dieser jedoch werkzeugbedingt und keine prinzipielle Schwäche, wird also in Zukunft beseitigt werden.

Nur einige Tools lassen sich in größerem Umfang erweitern: Westmount OMT und StP/OMT enthalten eine Programmiersprache, mit der Entwickler auf das Repository zugreifen können, um beispielsweise die Codegenerierung nach eigenen Bedürfnissen zu gestalten. Andere Funktionsbereiche (Projektmanagement, Anbindung an Dokumentationswerkzeuge, Testwerkzeuge und so weiter) unterstützen die PC-basierten Werkzeuge überhaupt nicht, für Unix-Produkte sind diese Fähigkeiten erst angekündigt oder nur in eingeschränktem Umfang erhältlich.

Unix-Tools stärker als PC-Lösungen

Zusammenfassend ist festzustellen, daß die Leistungsfähigkeit der objektorientierten Unix-Werkzeuge zwar stark differiert, die der PC-Lösungen im Moment jedoch deutlich übersteigt. Die untersuchten PC-Tools sind höchstens für eine Einarbeitung in objektorientierte Methoden einsetzbar.

Folgende Schlagwörter beschreiben aktuelle Tendenzen der Systementwicklung:

- Client/Server, verteilte heterogene Umgebungen,
- grafische Oberflächen,
- wachsende Bedeutung von Querschnittsfunktionen wie Qualitäts- und Projektmanagement,
- wachsende Bedeutung von Wiederverwendung und Reengineering,
- Geschäftsprozeßmodellierung.

Neben 'Objektorientierung' ist 'Client/Server' im Moment das Thema, das sich der größten Beachtung erfreut. Die mit diesem Begriff gemeinten Trends haben alle das Ziel, die vom

Methoden		besondere Vorgehensweisen	Generierung				PM	KM
konventionell	objektorientiert		DB-Schemata	Code	maskenor. UI	grafisches UI		
SA, SD		IE	●	●	●	●		
SA, SD, ERM				●		●		●
SA, ERM		Merise, SDMS			●	●		
ERM, RT		SA, SC	●				●	
ERM		IE	●	●	●	●		●
ERM, SA, SD			●	●	●			
ERM		V-Modell, SETec, Merise, SSADM	●	●	●		●	●
	OOA, OOD			●				
	OMT			●				
ERM, SA		Isotec						●
ERM, SA, SD								
SA, AD	OOA, OOD, OMT							●
ERM, SA, SD					●			●
	OOA, OOD	VSOM		●				
ERM, SA, SD		LBMS-SE	●	●	●	●	●	●
ERM, SA, SD	OOSA	V-Modell	●	●				●
ERM, SA, SD	OMT	Isotec, SSADM, V-Modell, (SETec)	●	●	●	●		●

OAOD: OO Analyse/Design (Coad/Yourdon)
OOA: Objektorientierte Analyse
OOD: Objektorientiertes Design
OOSA: OO Systemanalyse (Shlaer/Mellor)
OMT: Object Modelling Technique (Rumbaugh)

PM: Projektmanagement
RT: Real-Time
SA: Strukturierte Analyse
SC: Structure Charts
SD: Strukturiertes Design
SERM: Strukturiertes ERM

STD: State Transition Diagram
VSOM: Vorgehensmodell für semantisches Objektmodell

Quelle: Universität Köln, Lehrstuhl f. Wirtschaftsinformatik, Systementwicklung

Großrechner geprägten zentralen, zeichenorientierten Strukturen zu ersetzen. An ihre Stelle soll eine leistungsfähigere verteilte Verarbeitung im Netz mit grafischen Oberflächen treten.

Konventionelle CASE-Tools waren traditionell auf den Großrechner ausgerichtet und sind durch den Trend weg von dieser Technologie in Probleme geraten. Einige Hersteller wie zum Beispiel LBMS haben diesen Zusammenhang früh erkannt und bereits Anfang 1993 entsprechende Modifikationen an ihren Produkten zum Abschluß gebracht. Praktisch alle Konkurrenten haben zumindest angekündigt, sich auch in diese Richtung zu bewegen.

Client/Server verlangt nach Umstellung

Daraus resultierende Weiterentwicklungen bedeuten zum einen eine Veränderung an den zugrundeliegenden Methoden. Zum anderen wandelt sich die programmtechnische Ausrichtung erheblich: Die Werkzeuge müssen nun andere Programmiersprachen (Scriptsprachen oder 4GLs statt Cobol) unterstützen. Auch bekommt eine leistungsfähige Datenbankgenerierung für möglichst mehrere Zieldatenbanken ein größeres Gewicht.

Alle Produkte unterstützen im Moment lediglich eine einfache Verteilungsarchitektur, bei der Clients unter grafischen Oberflächen auf Datenbankserver zugreifen. Es ist derzeit nicht möglich, Programme zu entwickeln, die leistungsfähigere und ohne grundsätzliches Redesign der

Anwendungen änderbare Verteilungsarchitekturen (vergleiche CORBA der OMG) einsetzen. In dieser Hinsicht bleiben die zur Zeit erhältlichen objektorientierten Werkzeuge hinter guten konventionellen Client/Server-Ansätzen wie Systems Engineer zurück.

Nicht nur Technik sorgt für Innovation

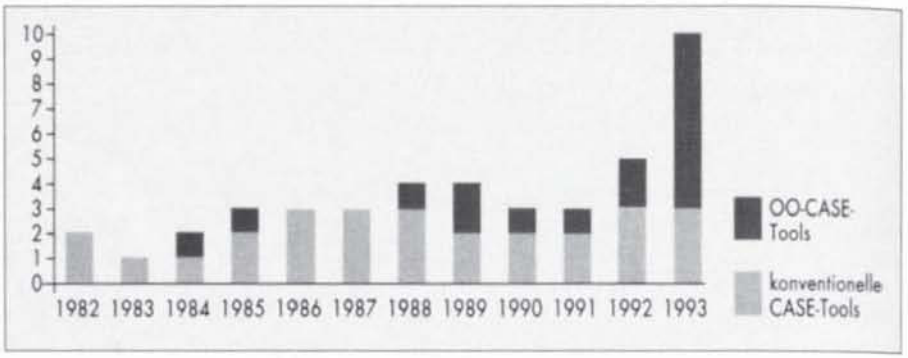
In der Vergangenheit wurde Produktion und Vertrieb von Software durch technologische Innovationen und die darauf aufbauenden neuen Anwendungsmöglichkeiten getrieben. Für die Nutzer von CASE bekommen betriebswirtschaftliche Aspekte wie die qualitäts- und zeitgerechte Auslieferung von Software eine immer größere Bedeutung. Daher werden phasenübergreifende Aktivitäten wie Qualitätssicherung und Projektmanagement immer wichtiger.

Dieser Tendenz kommen Werkzeuge mit breiter Funktionspalette wie Teamwork oder Maestro II oder erweiterbare Werkzeuge entgegen, die wie Innovator von MID um Test- oder Projektmanagementfunktionen ergänzt werden können. Objektorientierte Produkte bieten außerhalb der Analyse- und Designaktivitäten hier noch keine Unterstützung.

Aufgrund des hohen Anteils der Wartung an den Gesamtkosten von oft über 80 Prozent und des großen Wertes der realisierten Anwendungen hatte das Reengineering bereits bei konventioneller Entwicklung eine hohe Bedeutung.

Der zunehmende Druck, der durch Outsourcing und den Trend zu Standardsoftware entsteht, lassen das Thema Wiederverwendung zunehmend ins Blickfeld geraten. Objektorientierten Werkzeugen sagt man hier Verbesserungsmöglichkeiten nach: Alle Hersteller nehmen für ihre Systeme in Anspruch, Wiederverwendung zu unterstützen. Sie berufen sich dabei auf die Vererbung auf Code-Ebene, die sich aus der objektorientierten Programmierung automatisch ergibt.

Für den unternehmensweiten Einsatz sind weitergehende Mechanismen bei der Verwaltung und Integration von Klassenbibliotheken von Bedeutung: Augenblicklich enthalten CASE-Tools weder leistungsfähige Suchfunktionen zum Wiederauffinden von Klassen, noch ist eine Durchgängigkeit der Werkzeuge über alle Entwicklungsphasen gegeben. Zum Beispiel lassen sich Klassenbibliotheken nicht in Upper-CASE-Tools importieren, so daß sich bei zugekauften Klassenbibliotheken Probleme ergeben.



Markteinführungszeitpunkte der CASE-Tools: Doch kein Ende der Euphorie? Hersteller führten 1993 zehn neue Produkte ein. Die objektorientierte Systementwicklung hat hieran einen großen Anteil.

Die große Lücke zwischen OO-Upper-CASE-Tools (Westmount I-CASE, StP und so weiter) und Entwicklungsumgebungen der Compiler-Hersteller für die Realisierungsphase führt dazu, daß die Verwaltung von Klassenbibliotheken über den gesamten Softwarelebenszyklus und die Wartungsphase bisher nur unzureichend unterstützt werden. Dadurch kommen wesentliche Vorteile der objektorientierten Systementwicklung wie Wiederverwendung und eine adäquate Unterstützung iterativer Systementwicklung nicht voll zur Geltung.

Im Zuge der Optimierung von Geschäftsprozessen wird dieses Thema für CASE relevant. Hier stehen die Hersteller objektorientierter Werkzeuge vor ungelösten Aufgaben, wenn sie eine Durchgängigkeit von der Geschäftsprozeßmodellierung bis zur Realisierung ermöglichen wollen. Pilotprojekte zeigen derzeit Probleme an der Schnittstelle zur objektorientierten Analyse auf. Von den untersuchten Programmen bot lediglich SOM-CASE zum jetzigen Zeitpunkt Ansätze für diesen Bereich.

Ältere Werkzeuge mit mehr Erfahrung

Konventionelle CASE-Tools sind objektorientierten im Moment überlegen. Aufgrund ihrer längeren Präsenz am Markt decken sie einen weiteren Funktionsbereich ab und sind an aktuelle Entwicklungstendenzen der Systementwicklung meist gut angepaßt.

Prinzipiell bieten objektorientierte Werkzeuge in methodischer und programmtechnischer Hinsicht das größere Potential. Sie sind aber, selbst wenn man den Aussagen der Hersteller Glauben schenkt, den konventionellen Werkzeugen in diesem Zusammenhang bis circa Ende des Jahres unterlegen.

Heutiger Stand der Werkzeugtechnik ist es, daß die konventionellen Produkte mit Editoren für die Beschreibungsmittel der objektorientierten Methoden versehen wurden, ohne deren Vorteile besonders zu unterstützen. Die so entstehenden Werkzeugfamilien sind häufig noch nicht integriert, so daß die objektorientierten Varianten nur beschränkten Zugriff auf Basisfunktionen der bisherigen konventionellen Produkte haben. Die besten CASE-Tools stammen daher mit wenigen Ausnahmen von den Herstellern, die schon Know-how im konventionellen Bereich gesammelt haben.

Eine große Lücke zwischen objektorientierten Upper-CASE-Tools und Entwicklungsumgebungen für die Realisierungsphase verhindert eine Verwaltung von Klassenbibliotheken über den gesamten Softwarelebenszyklus und eine weitergehende Unterstützung der Wartungsphase. Neben der Unvollständigkeit der Werkzeuge ist diese Lücke zur Zeit die Ursache dafür, daß wesentliche Vorteile der Objektorientierung nicht voll zur Geltung kommen. Die Hersteller haben diese Defizite erkannt. Für das laufende Jahr kündigen sie teilweise bis zu vier neue Versionen ihrer Produkte an. (bl)

MICHAEL KUNZ,
JAN GEISSELMANN,
ANDREAS HIERHOLZER,
GEORG HERZWURM

sind Mitarbeiter am Lehrstuhl für Wirtschaftsinformatik, Systementwicklung, der Universität Köln und beschäftigen sich mit Softwarequalitätsmanagement.