CASE-Tools -
Perspektivenwandel am Weg vom Anfänger zum Experten

E. Hochmüller, C. Kohl, H.C. Mayr, R. Mittermeir

Für ein bestimmtes phasenübergreifendes CASE-Tool kann man sich eigentlich nur entscheiden, wenn man zuvor mit den in Frage kommenden Werkzeugen entsprechende Erfahrungen gesammelt hat. Dies ist in der Praxis kaum möglich. Im universitären Umfeld bieten sich dagegen bessere Möglichkeiten für einen umfassenden Werkzeugvergleich. Wir berichten hier über Erfahrungen aus einer Studie, in der mit verschiedenen Werkzeugen die selbe Problemstellung bearbeitet wurde. Dabei gehen wir sowohl auf die beobachteten Anfängerprobleme als auch auf solche Aspekte ein, die für den versierten Benutzer relevant sind.

Motivation

Im breiten Markt integrierter CASE-Tools ist es mühsam, die Übersicht zu bewahren. Kriterienkataloge (z.B. [duPlessis 93], [Church/Matthews 95]) helfen ein wenig, Durchblick zu gewinnen, doch es bleiben eine Fülle von Fragen. Etwa: Wie sind diese Kriterien entstanden? Viele scheinen selbstverständlich! Welche sind für meine Situation von höherer bzw. von geringerer Bedeutung? Erfüllen die Werkzeuge, die in engerer Wahl stehen, laut ihrer Beschreibung alle mir wichtig erscheinenden Kriterien? In welchem Ausmaß tun sie das? Ist die Werkzeugentscheidung nicht gleich der richtige Zeitpunkt, die Entwicklungsmethodik zu ändern? Sollte ich nicht ... ?

Alle diese Fragen können letztlich nur individuell beantwortet werden. Hat man sie beantwortet, wird die getroffene Entscheidung hoffentlich zur sich selbst erfüllenden Prognose. Die Investitionskosten sind in der Regel hoch genug, um nicht bei nächster Gelegenheit einen Wechsel des Werkzeugs vorzunehmen. - Selbst dann nicht, wenn sich gelegentlich ein leiser Fluch durch die Zähne der Entwickler preßt. Man wechselt nicht mehr. Auch dann nicht, wenn die Flüche nicht nur leise und heimlich sind, sondern das angeschaffte Werkzeug durch eine Revolution der Entwickler abgelehnt wird. Tritt dieser Fall ein, landet die Dokumentation im Schrank und die Software auf einem Archivband. Ab diesem Zeitpunkt weiß man es: “CASE ist eine leere Worthülse! - Ein neues Tool kommt uns so schnell nicht mehr ins Haus.”

Diese Situationsbeschreibung schien uns typisch. Typisch sowohl für den erfolgreichen als auch für den nicht erfolgreichen Beginn mit einem CASE-Werkzeug. Wie sorgsam das Auswahlverfahren auch sein mag, ein paralleler Einsatz mehrerer komplexer Werkzeuge in echten Entwicklungsprojekten ist kaum jemals gegeben. Deshalb versuchten wir, diese Fragestellungen durch ein vergleichendes Experiment in unserer universitären Umgebung zu untersuchen. Wir bemühten uns dabei allerdings, die Fallgruben typischer Studenten-Experimente nach Möglichkeit zu vermeiden. Ein weiteres Ziel war es, mehr Klarheit und Struktur in diejenigen Eigenschaften von Softwareentwicklungsmethoden und -werkzeugen zu bringen, die für eine Auswahlentscheidung bedeutsam sind.

Nachfolgend stellen wir einige der im Rahmen des vergleichenden Experiments gewonnenen Erfahrungen vor.

Experimentaufbau

Als organisatorischer Rahmen für das Experiment diente ein im Studienplan vorgeschriebenes Praktikum. Das Experiment wurde in zwei Durchgängen während zweier aufeinanderfolgender Semester gestartet, wobei sich jeder Durchgang über nahezu zwei Semester erstreckte.

Im ersten Durchgang wurden "konventionelle CASE-Tools" analysiert. Es waren dies IEW™, Innovator™, Oracle*CASE™ und ProMod-PLUS™ [Mittermeir et al. 93]. Im zweiten Durchgang wurden Methoden und Werkzeuge zur Unterstützung objektorientierter Software-Entwicklung, nämlich OMTool™, ObjectMaker™, Objectory™, Rational Rose™, ObjectiF™ und SOMTool™ [Kaschek et al. 95], untersucht. Die Auswahl der konventionellen Werkzeuge erfolgte nach Reifegrad und Marktpräsenz - Kriterien, die bei den objektorientierten Werkzeugen noch nicht zum Einsatz gelangen konnten. Hier wurde vielmehr getrachtet, auch das Methodenspektrum einer Beurteilung zu unterziehen.

An der Evaluation eines Werkzeuges arbeitete jeweils ein Team von drei Studierenden, wobei jeder von ihnen durchschnittlich 400 bis 500 Stunden einbrachte. Dies entspricht etwa 10 Personenmonaten pro Team. Jedem Team war ein betreuender Assistent zugeordnet, der die Rolle des Fachvorgesetzten übernahm. Ferner waren die innerhalb eines Experiment-Durchlaufes arbeitenden Teams einem Professor berichtspflichtig. Dieser spielte mithin - auf Industriemuster übertragen - die Rolle einer Führungsperson zweiter Ebene.

Als Anwendungsbeispiel für das Experiment wurde die Erstellung eines Konferenz-Verwaltungssystems gewählt, eine Aufgabenstellung, für die in den IFIP-CRIS-Studien [Olle 88] eine Fülle von Musterlösungen und insbesondere eine klare Problembeschreibung vorliegt. Diese Vorgaben wurden dahingehend erweitert, daß neben der Unterstützung statischer Aspekte auch die dynamischer Aspekte in ausreichendem Ausmaß untersucht werden konnte.

Zu den Studierenden ist zu ergänzen, daß alle Teilnehmer bereits je eine Lehrveranstaltung über Datenbank-Entwurf und Software-Entwurf (beides mit Übungen) besucht hatten, in deren Rahmen sie in konzeptueller Modellierung mittels ER, sowie in OMT- und OOA-Modellierung aber auch in SA/SD ausgebildet worden waren. Soweit die Werkzeuge Standard-Methoden unterstützten, waren diese Methoden und Notationen mithin bereits bekannt.

Um in diesem Experiment über Aussagen zu typischen Anfängerproblemen hinauszukommen und für entsprechenden Erfahrungsaustausch (und Lerngewinn) der einzelnen Teams zu sorgen, wurde nach folgendem Ablaufplan vorgegangen:

1. Erläuterung der Problemstellung und Zuordnung zwischen Team, Betreuer und Werkzeug.

2. Einarbeitung des Teams in das Werkzeug mit Unterstützung des Betreuers (bei objektorientierten Ansätzen war teilweise auch eine Einarbeitung in eine bislang unbekannte Entwurfsmethode erforderlich). Diese Einarbeitung erfolgte anhand der

für das betreffende Tool vorgesehenen Dokumentation und allfälliger Tutoriumsbeispiele. Als Schwerpunkt dieser Phase, für die etwa ein Viertel der Gesamtprojektdauer vorgesehen war, sollte ein "typisches Studentenbeispiel" ausgearbeitet und so erste Erfahrungen mit dem Werkzeug gemacht werden.

3. Danach hatte in einer Plenarsitzung zunächst jedes Team sein Werkzeug und seine Erfahrungen und Probleme während der Lernphase zu präsentieren. Anschließend wurde den Studierenden die Problemstellung für das eigentliche Projekt, Entwurf (und Erstellung) von Software für die Organisation einer Konferenz, vorgestellt.

4. In der nun folgenden Projektphase mußte jedes Team die Analyse für das Anwendungsproblem durchführen. "Kunde" war dabei jeweils ein Team-Betreuer. Um auch die Unterstützung von Modell-Änderungen durch das jeweilige Werkzeug zu analysieren, wurde vereinbart, daß sich die Anforderungen des Kunden zwischen Erstinterview und Abnahme in gewissen Belangen zu ändern hätten. Nach einem Zwischenreview war mit Werkzeugunterstützung ein Entwurf anzufertigen. Einige Teams konnten (sofern vom Werkzeug überhaupt angeboten) auch die Implementierungsunterstützung ansatzweise erproben. Die jeweiligen Ergebnisse waren sauber zu dokumentieren.

5. Das Experiment endete mit einer Schlußpräsentation im Plenum. Dort waren die Ergebnisse vorzustellen (umfangreicher Design-Review) und über die Erfahrungen mit dem Werkzeug zu berichten.

Beobachtungen zu den Methoden

Jedes der untersuchten Werkzeuge fußt auf einer bestimmten Methode oder unterstützt ein spezifisches Methodenspektrum. Dennoch ergaben sich zwischen den beiden Durchgängen des Experiments unter diesem Aspekt wesentliche Unterschiede.

Die konventionellen Werkzeuge unterstützen weitgehend eine einheitliche Entwurfsmethodik, nämlich die Erstellung eines ER-Modells als Statik-Modell, auf dessen Basis ein SA-Datenfluß-Modell als Dynamik-Modell entwickelt wird. Im objektorientierten Bereich sind die angebotenen Konzepte für Statik- und Dynamik-Modellierung dagegen teilweise recht unterschiedlich. Um dem Rechnung zu tragen, wurden hier Methodeneigenschaften der Auswahl konkreter Werkzeuge vorangestellt. Zu diesem Zweck wurde ein eigener Katalog von Methoden-Charakteristika erarbeitet [Kaschek et al. 95]. Bei der konkreten Werkzeugauswahl wurde darauf geachtet, daß jeweils verschiedene, jedoch bekannte Methoden aus dem breiten Spektrum der objektorientierten Analyse und des objektorientierten Entwurfs behandelt werden (OMT, OOA, SOM, OOSE, Booch, RDD).

Manche Teams wurden demzufolge auch mit ihnen bislang unbekannten Methoden konfrontiert. Ihnen stellte sich somit die zusätzliche Aufgabe, sich vor den Werkzeugaspekten auch eine neue Methode und die dort existierenden Perspektiven auf ein zu modellierendes System sowie entsprechende Denkmuster anzueignen. Die Dokumentation dieser Methoden war teilweise in Form von Monographien gut aufbereitet, teilweise jedoch nur bedingt für ein Selbststudium geeignet. In diesen Fällen waren auch die Werkzeug-Handbücher nicht sehr hilfreich, da sie großteils als reine Menü-Bedienungsanleitung konzipiert waren und nicht die

Philosophie der zugrundeliegenden Methode präsentierten.

Bei der Erarbeitung des neuen Methodenwissens zeigte sich, daß den Projektteams, verwöhnt durch die semantische Mächtigkeit von OMT bzw. OOA, eine der neu im Selbststudium zu erlernenden Methoden im Vergleich zu bekannten Methoden nicht mächtig genug erschien (RDD). Die Einarbeitung in die Methode von Booch erwies sich als verhältnismäßig unproblematisch, wogegen OOSE in seinen Phasen nicht als logisch und bruchfrei ermpfunden wurde. Als besonders schwierig erwies sich die Einarbeitung in SOM, das eine im Vergleich zu bekannten - eher an Domain-Objekten orientierten Philosophien - eine ganz andere, prozeßorientierte Denkweise verlangte. Außerdem erschien die Methode nur lückenhaft dokumentiert.

Vor allem in der Praxis ist die Wahl des richtigen Werkzeugs entscheidend. Oft wird jedoch der Fehler begangen, sich von einem gut geschulten Verkäufer überzeugen zu lassen, daß das von ihm angebotene Tool allein die Lösung zu besserer Produktivität und Qualität der zu erstellenden Software sei. In diese Falle tappt man leicht hinein, wenn man von einem CASE-Tool erwartet, daß es den Entwicklern die Arbeit abnimmt. So garantieren beispielsweise auch Textverarbeitungsprogramme allein nicht, daß die mit ihnen verfaßten Texte literarische Reife haben. Befaßt man sich hingegen zunächst mit dem eigenen Softwareentwicklungsprozeß und den dabei idealerweise anzuwendenden Methoden, dann sollte man auch ein Tool finden können, das sich in diesen Entwicklungsprozeß einbinden läßt. Fällt die Wahl auf eine objektorientierte Vorgangsweise, muß man sich jedoch bewußt sein, Pionierarbeit zu leisten. Objektorientierte Ansätze sind eben relativ jung und inhomogen und können somit den Reifegrad, wie ihn etwa ER oder SA/SD erreicht haben, noch nicht aufweisen.

Einarbeitung der Teams - Überwinden der Anfangsprobleme

"Aller Anfang ist schwer"- dies bewahrheitete sich im ersten Umgang mit den verschiedenen CASE-Tools. So waren speziell in dieser Anfangsphase folgende Hürden zu bewältigen:

Schulung

Die am Projekt beteiligten Studierenden und Betreuer konnten aus Kostengründen bis auf eine Ausnahme (ProMod-PLUS) keine Herstellerschulungen in Anspruch nehmen. So erforderten bereits kleinere Verständnisprobleme, die durch einen Tool-Experten leicht aus dem Weg zu räumen gewesen wären, relativ großen Zeitaufwand. "Tips und Hints" von Tool-Spezialisten, die die Tücken ihrer Systeme ja kennen sollten, hätten den Teams auch später projektbegleitend diverse schlaflose Nächte erspart.

In der Praxis erfolgt die Einführung des zukünftigen CASE-Teams in Methode und Werkzeug üblicherweise durch vom Hersteller selbst angebotene mehrtägige Schulungen. Dabei stehen meist die Eigenschaften des Werkzeuges im Vordergrund, welche anhand von Tutorials gemeinsam durchgespielt werden. Die methodische Weiterbildung wird durch das Werkzeug bestimmt, sodaß das Verständnis der Vor- und Nachteile der Methode von den Schulungsteilnehmern sehr subjektiv gesehen wird. Idealerweise stehen bei Unklarheiten bzw. Anwendungsschwierigkeiten den Neulingen für einen gewissen Zeitraum ständig Experten für Rückfragen zur Verfügung. Mittel- und längerfristig sollte man anstreben, erfahrene eigene Mitarbeiter für Schulungen einzusetzen. So kann sichergestellt werden, daß auch die Tücken

des jeweiligen Tools von Beginn an adäquat behandelt werden.

Methodenvorwissen

Beim ersten Bekanntschaftschließen mit Werkzeugen, welche auf einer den Studierenden bekannten Methoden basierten, war der Einarbeitungsaufwand relativ gering. Die einzelnen Teams wurden rasch auf die offensichtlichsten Unzulänglichkeiten aufmerksam, die stark hervorgehoben und teilweise überbewertet wurden. Länger dauerte die Anfangsphase derjenigen Teams, die ihnen bislang unbekannte Methoden verwenden sollten, was aber nicht nur den einzelnen Werkzeugen zugeschrieben werden kann.

Es wurde von einigen Teams auch versucht, eine bisher unbekannte Methode mit Hilfe des dazu zur Verfügung stehenden Werkzeuges zu erlernen, was jedoch ohne echtes Verständnis der Methode scheitern mußte. Das Vorgehensmodell der Methoden war nämlich in keinem der Werkzeuge ausreichend implementiert und mußte im Kopf der Benutzer existieren, um das Werkzeug sinnvoll einsetzen zu können. Das “Rätsel von SERM” konnte beispielsweise erst durch das Anfordern einer Habilitationsschrift gelöst werden. Das primäre Werkzeugstudium war jedoch dann gerechtfertigt, wenn die betreffende Methode vor allem durch das Werkzeug und mittels der dort verfügbaren Beispiels-Entwürfe dokumentiert war (SOM).

Einstiegsunterstützung

Basis für die Einarbeitung in die Werkzeuge waren die zur Verfügung stehenden Handbücher, Tutorials und Methoden-Dokumentationen. Insbesondere Musterbeispiele, welche in den Handbüchern ausreichend dokumentiert waren, erwiesen sich als hilfreich. Als Ansprechpartner bei Problemen standen die Team-Betreuer und darüberhinausgehend meist “Hotlines” zu den Herstellerfirmen zur Verfügung, die aber nur im Notfall beansprucht wurden.

Methodenabweichungen im Detail

Gemeinsam war allen Werkzeugen, daß keines nach der Einarbeitungsphase den Erwartungen voll entsprach und jedes Team lieber ein anderes, besseres Werkzeug hätte verwenden wollen. Besonders bei den objektorientierten Werkzeugen stellte sich heraus, daß diese das von der jeweiligen Methode her vorgesehene Modellierungssystem nicht vollständig zur Verfügung stellten. Auch die Gestaltung der Benutzeroberflächen wurde teilweise kritisiert.

Anfängerfehler

Wie generell bei anfänglicher Verwendung von Anwendungssystemen, so traten auch beim hier geschilderten Experiment typische Probleme auf, die zwar anfangs als unüberwindliche Hürden erschienen, jedoch im Laufe der Zeit durch entsprechenden Umgang mit dem Tool vermieden werden konnten. Zum einen erlernten die Studenten die korrekte Handhabung des jeweiligen Werkzeuges, zum anderen mußten (teilweise trickreiche) Strategien zur Überbrückung von Unzulänglichkeiten des Werkzeuges entwickelt und angewandt werden. So zum Beispiel machte eine nicht zur Verfügung gestellte “undo”-Operation das Entwicklerleben unnötig schwer.

Feedback

“Hilfe! - Unser Werkzeug ist das schlechteste!” -- “Oder?” -- “Jene der anderen Teams sind auch nicht so toll, wie wir meinten, als wir am Gang mit den Kollegen sprachen.” Dieser Meinungswandel ist eine interessante Invariante, die sich nach der Einarbeitungsphase bei allen Teams zeigte, die mit vergleichbaren und etablierten Methoden arbeiteten. In unserem Experiment führte diese pauschale Unzufriedenheit wohl nicht zu wirklichen Problemen, da das Experiment zwar eine wesentliche Erfahrung für die Studierenden darstellte, jedoch nur als ein kleiner Stein am Weg ihrer weiteren Entwicklung aufgefaßt werden kann. In der industriellen Praxis kann die Tatsache, daß Erwartungshaltung und Ersterfahrung zu sehr auseinanderklaffen, jedoch zum k.o.-Kriterium werden. Entwickler lehnen in diesem Fall das angeschaffte Werkzeug gänzlich ab. Der Versuch einer Wiedergutmachung durch Einführung eines anderen Werkzeugs ist unter diesen Umständen vielfach durch die bisherigen negativen Erfahrungen von Anfang an zum Scheitern verurteilt.

Mit der Erfahrung wachsen die Wünsche

Erfuhren die Teilnehmer des Experiments bei der Zwischenpräsentation, daß nicht nur sie mit dem “schlechtesten” Tool zu arbeiten hatten, so zeigte sich am Projektende eine neue Invariante. Alle “Tester” herkömmlicher CASE-Tools antworteten auf die Frage, ob sie in ihrem nächsten Softwareentwicklungsprojekt wieder ein CASE-Tool einsetzen würden, einstimmig mit “ja”. Auf die Zusatzfrage nach dem Tool, das sie dabei einsetzen würden, wählte jede(r) Teilnehmer(in) das bereits bekannte “eigene” Tool. Nahezu alle Teams meinten, daß sie bei Durchführung des nächsten Projektes vergleichbarer Größenordnung wieder zu dem Werkzeug greifen würden, mit dem sie ihr “Gesellenstück” angefertigt hatten. Es bedarf schon gravierender Mängel am eigenen Werkzeug, um einen Werkzeugwechsel in Erwägung zu ziehen. Offenbar haben Abschreibungsmodelle nicht nur im kaufmännischen Bereich ihren Platz. Man möchte auch Lern- und Einarbeitungsinvestitionen amortisiert haben, bevor man sich auf neue einläßt.

Im Falle von objektorientierten CASE-Werkzeugen blieb man jedoch nicht nur weitestgehend bei der anfänglichen Aussage, sondern witzelte sogar, daß man hier nicht von CASE sondern von CDSE (Computer Documented Software Engineering) sprechen sollte. Objektorientierte CASE-Tools haben eben schlichtweg den Reifegrad der herkömmlichen CASE-Werkzeuge noch nicht erreicht und sind hauptsächlich im Upper-CASE-Bereich angesiedelt. Somit ist es nicht verwunderlich, wenn diese Tools mit einem elektronischen Bleistift verglichen werden, mit dem es sich leichter radieren und replizieren läßt, als ohne Verwendung eines solchen Hilfsmittels.

Unabhängig von den einzelnen Werkzeugen bleiben einige Grundprobleme, die je nach Werkzeug in unterschiedlicher Betonung als dringliche Verbesserungswünsche angeführt werden müssen:

Konsistenz- und Integritätsunterstützung

Durch das Überprüfen und Nachziehen von Konsistenz- und Integritätsbedingungen unterscheiden sich CASE-Tools von reinen Zeichenwerkzeugen mit vordefinierten Elementen. Diese Einstiegsprüfung wurde mittlerweile von den meisten klassischen CASE-Werkzeugen

in ihren neuesten Releases bestanden. Viele objektorientierte CASE-Tools haben diese Hürde jedoch noch nicht bewältigt.

Änderungsfreundlichkeit

Besonders bei phasenübergreifenden CASE-Werkzeugen ist das Zusammenwirken der einzelnen Toolkomponenten eine zusätzliche Notwendigkeit, die solche Tools eindeutig von herkömmlichen Zeichenwerkzeugen abhebt. Hier schnitten die konventionellen CASE-Werkzeuge ebenfalls eindeutig besser ab als objektorientierte Tools, auch wenn sie je nach Methode verschiedene Modellierungskomponenten anbieten, die jedoch nur lose miteinander gekoppelt sind. Somit bleibt es dem Entwickler selbst überlassen, bei Änderungen in einer Darstellungsart, diese auch in den jeweils anderen "verwandten" Teilmodellen nachzuziehen. Wenn man weiters bedenkt, daß "echte" Projekte nicht immer lehrbuchmäßig von Phase zu Phase fortschreiten, sollte von einem durchgängigen CASE-Tool wohl zu erwarten sein, daß die Änderung einmal getroffener Entscheidungen in geeigneter Weise unterstützt wird.

Teamfähigkeit

Da die Erstellung von Software-Systemen grundsätzlich in Teams erfolgt, waren wir über Defizite überrascht, die die Werkzeuge auf diesem Gebiet aufwiesen. Dies gilt gleichermaßen für konventionelle und objektorientierte Tools.

Druckmöglichkeit

So komfortabel die Benutzeroberfläche bei vielen Tools auch gestaltet ist, so enttäuscht waren wir über die mangelhafte Unterstützung der Berichtserstellung. Ab einer gewissen Größenordnung lassen sich manche Entwürfe nur mehr mit der Lupe betrachten. Ausgaben auf mehreren, später zusammenklebbaren Blättern werden nur von sehr wenigen Tools angeboten. Vielmehr muß man sich mit "Screenshots" behelfen, um die Arbeiten einigermaßen lesbar auf Papier bringen zu können.

Zuammenfassung

Neben vielen Detailergebnissen, die wohl schwer generalisierbar sind, können wir als Hauptergebnis auf die kritische Phase des Ersteinsatzes in einem Produktionsprojekt hinweisen. Umso geringer Erwartungen und deren Erfüllung auseinanderklaffen und umso besser dieser Ersteinsatz durch geeignete Schulungs- und Begleitmaßnahmen abgestützt wird, desto höher ist die Wahrscheinlichkeit, mit dem gewählten Werkzeug zufrieden zu sein. Ist diese kritische Phase allerdings einmal überwunden, sichert menschliche Latenz den weiteren Erfolg des gewählten Produktes, solange dieses die für die spezifische Anwendungssituation wesentlichsten Kriterien erfüllt.

Zudem ist festzuhalten, daß bei Werkzeugen, die klassische Softwareentwicklung unterstützen, ein gewisser Grundkonsens bezüglich der zu unterstützenden Bereiche gegeben erscheint. Bei objektorientierten Werkzeugen ist dieser Reifegrad aufgrund des Nachholbedarfs der zugrundeliegenden Methoden allerdings noch nicht erreicht.

Referenzen

[Church/Matthews 95]
Church T., Matthews P:: An Evaluation of Object Oriented CASE Tools: The Newbridge Experience, Proc. Seventh International Workshop on Computer Aided Software Engineering (CASE 95), Toronto, 1995, pp. 4-9

[duPlessis 93]
duPlessis A.L.: A method for CASE tool evaluation, Information & Management, Vol. 25, No. 2, Aug. 1993, pp. 93-102

[Kaschek et al. 95]
Kaschek R., Mayr H.C., Kohl C., Kop C.: Characteristics of OOA Methods and Tools - A First Step to Method Selection in Practice, Technical Report 95/1, University of Klagenfurt, 1995

[Mittermeir et al. 93]
Mittermeir R., Hochmüller E., Kienzl K., Kofler E., Steinberger H.: CASE-Tool Evaluation in einem Multi-Development Projekt, Proc. ADV-Tagung CASE 93, Wien, Okt. 1993, pp. 6-26

[Olle 88]
Olle T.W.: System Design Specifications for a Conference Organization System, in Olle T.W., Verrijn-Stuart A.A., Bhabuta L. (eds.): Computerized Assistance During the Information System's Life Cycle, North-Holland, 1988, pp. 497-539

Danksagung

Wir danken den Distributoren der eingesetzten Software für die im Zusammenhang mit diesem Evaluationsprojekt gegebene Unterstützung.