CASE-Tool Evaluation in einem Multi-Development Projekt

R. Mittermeir, E. Hochmüller, K. Kienzl, E. Kofler, H. Steinberger
Institut für Informatik
Universität Klagenfurt

Dipl.-Ing.Mag.Dr. Roland Mittermeir

Roland Mittermeir studierte an der Technischen Universität Wien Informatik und an der Wirtschaftsuniversität Wien Betriebswirtschaft. Nach Tätigkeit am Institut für Höhere Studien Wien, der Technischen Universität Wien, der University of Texas at Austin und der University of Maryland at College Park wurde er 1984 an die Universität Klagenfurt berufen, wo der das Institut für Informatik aufbaute.

Im Bereich der Lehr-, Forschungs- und Beratungstätigkeit konzentriert sich Professor Mittermeir innerhalb der Angewandten Informatik insbesondere auf die Bereiche Requirements Engineering, Entwurf von Anwendungssystemen, Sprachentwurf und Systementwicklungsmethodologie.

Dr. Elke Hochmüller, Dipl.-Ing. Klaus Kienzl, Mag. Evelin Kofler und Dr. Hannes Steinberger sind Assistenten am Institut für Informatik der Universität Klagenfurt. Ihre jeweiligen Forschungsinteressen liegen in den Bereichen Software-Reuse und CASE-Tools, Software-Restrukturierung, Software-Spezifikation sowie Endbenutzersprachen und Software-Metriken. Sie leiteten während des hier beschriebenen Projektes jeweils eine Evaluationsgruppe.

CASE-Tool Evaluation in einem Multi-Development Projekt

R. Mittermeir, E. Hochmüller, K. Kienzl, E. Kofler, H. Steinberger
Institut für Informatik
Universität Klagenfurt

ABSTRACT

Im Wintersemester 1992/93 wurde im Rahmen des Praktikums aus Betriebsinformatik ein "Vergleichs-Wettkampf" zwischen den CASE-Produkten IEW, Innovator, Oracle*CASE und ProMod-PLUS durchgeführt.

Das Projekt war so gestaltet, daß jeweils studentische Kleingruppen unter Anleitung eines wissenschaftlichen Mitarbeiters ein System zur Unterstützung der Abwicklung internationaler Tagungen entwerfen (und nach Maßgabe der Tool-Unterstützung partiell entwickeln) mußten. Durch Komplexität und Facettenreichtum des Anwendungsproblems wie durch entsprechende Vorlaufschulungen konnten die eingesetzten Werkzeuge tatsächlich weitestgehend ausgelotet und einige für den Einsatz in Praxis und Lehre interessante Ergebnisse erzielt werden.

Im folgenden Vortrag sollen erste Ergebnisse aus diesem Projekt der Öffentlichkeit vorgestellt werden.

1. MOTIVATION

Der Einsatz von CASE-Tools wird zusehends zu einer notwendigen Voraussetzung für die moderne Software-Erstellung. Nicht, daß man qualitativ hochstehende Software nicht auch ohne diese Werkzeuge effizient erzeugen könnte - die Randbedingungen der Erstellung sauberer Software unter knappen Zeitbedingungen durch durchschnittliches Software-Entwicklungspersonal erzwingen jedoch nahezu, daß Werkzeuge eingesetzt werden, die Entwickler zu sauberem Entwurf, modularer Strukturierung sowie vollständiger und korrekter Dokumentation führen. Da darüberhinaus durch den Einsatz von CASE-Tools in der Software-Entwicklung auch Produktionsvorteile, vor allem aber Standardisierungsvorteile erzielt werden, ist heute nahezu jeder Software-Entwickler gehalten, insbesondere jene Software, die in Mehrpersonen-Teams entsteht, mit Hilfe von CASE-Tools zu erstellen.

Aufgrund dieser Randbedingungen stellte sich für uns als Universitätsinstitut, das im Bereich Software-Engineering tätig ist, die Frage,

- wie weit wir CASE-Tools aus Speziallehrveranstaltungen heraus in das Standardangebot einbinden können,
- auf welchen CASE-Tools diese Ausbildung abgestützt werden sollte,
- welche CASE-Tools wir Praktikern mit bestem Gewissen empfehlen können.

Die hier vorliegende Arbeit soll einen kurzen Überblick über das entsprechende Evaluationsprojekt bieten und dabei insbesondere CASE-Tool-Einsteigern eine Hilfestellung für

anstehende Auswahlentscheidungen (und Einführungsstrategien) bieten. Da die in diesem Projekt gewonnenen Erfahrungen hier nicht in aller Tiefe ausgeführt werden können, aber gerade Frustrationen zu Beginn eines derartigen Projektes oft zu einem langfristigen Scheitern führen, beschränken wir uns darauf, hier eine Negativ-Hitliste vorzustellen. Ein Vorschlag für positive Auswahl-Kriterien ist sehr stark von der konkreten Situation des jeweiligen Software-Hauses (bzw. Eigenentwicklers) abhängig. Eine detaillierte Diskussion eines solchen Kriterienkatalogs wird einer weiteren Publikation [MITT 93] vorbehalten bleiben.

Zum Untersuchungsgegenstand dieses Projekts ist festzustellen, daß wir CASE-Tools im klassischen Sinn analysierten, also Werkzeuge, die sich in Anlehnung an Phasenmodelle der Software-Entwicklung mit der Unterstützung von Analyse, Entwurf und Codierung und der Dokumentation dieser Schritte auseinandersetzen. Diese klassische (eingeschränkte) Sichtweise entspricht durchaus der gegenwärtigen Situation am Tool-Markt und der Einsatzhäufigkeit von Werkzeugen zur Software-Erstellung, wenn man von jenen Komponenten wie Compiler und Debugger, die heute als unabdingbare Programmierhilfen angesehen werden, absieht. Spezialwerkzeuge, wie etwa Spezifikations- Tools, Test-Tools oder Re-Engineering Tools hätten somit weder in das Vergleichs-Projekt gepaßt noch werden sie in der Software-Entwicklungslandschaft zum gegenwärtigen Zeitpunkt großflächig eingesetzt.

In dieser Arbeit schildern wir zunächst den Aufbau und die Rahmenbedingungen des Experiments und berichten anschließend über die Hürden, die sich einerseits CASE-Anfängern anderseits aber auch erfahreneren "CASE-Entwicklern" entgegenstellen. Darauf aufbauend wird ein Resumee über die insgesamt in diesem Projekt gewonnenen Erfahrungen gegeben, wobei dieses Kapitel auch eine Wunschliste an CASE-Anbieter enthält.

2. AUFBAU DES EXPERIMENTS

2.1. Rahmenbedingungen

CASE-Tools sind komplexe Software, die nicht durch bloßes Hinsehen auf ihre Zweckmäßigkeit für bestimmte Anwendungsspektren beurteilt werden kann. Auch kleine Spielzeug-Projekte vom Umfang eines halben bis ganzen Entwickler-Tages geben wenig Aufschluß über die Tücken von Werkzeug und Materie. Um "das Biest" wirklich kennen zu lernen und somit faire und verläßliche Aussagen treffen zu können, bedarf es der Durchführung eines realen Projektes mit dem entsprechenden Werkzeug.

Dem stellen sich in der Praxis allerdings erhebliche Hindernisse entgegen, wenn ein solches reales Projekt letztlich oder primär der Werkzeug-Evaluation dienen sollte. Einschulungsaufwand und -kosten sind viel zu hoch, um eine Gruppe in mehreren Werkzeugen schulen zu lassen und dann nach Durchführung jeweils eines Projektes im Umfang mehrerer Mannmonate eine Entscheidungsgrundlage zu geben. Weiters würden, wenn dasselbe Projekt mit mehreren Werkzeugen ausgeführt wird, Lerneffekte eintreten, die einem fairen Tool-Vergleich entgegenstehen. Das zuerst eingesetzte Tool wäre in einem solchen Experiment von vornherein nahezu chancenlos. Läßt man demgegenüber unterschiedliche Werkzeuge durch unterschiedliche Teams beurteilen, so wird diese Beurteilung realistischerweise in unterschiedlichen Projekten erfolgen müssen. Damit ist aus der Sicht

des Gegenstandsbereiches keine Vergleichbarkeit mehr gegeben. Weiters - und dies entspricht auch der in unserem Projekt gemachten Erfahrung - würden die einzelnen Gruppen trotz aller entdeckten Mängel (mit Recht) lieber mit dem "eigenen" Tool weiterarbeiten, als einen radikalen Toolwechsel mitvollziehen. Somit ist auch aus Sicht der innerbetrieblichen Zufriedenheit einem sauber durchgeführten Toolvergleich mit Skepsis zu begegnen.

In der Lehrveranstaltung "Praktikum aus Betriebsinformatik" mußten diese Aspekte nicht berücksichtigt werden.

Das Praktikum wird von viert- oder fünft-semestrigen Informatikstudenten besucht, die am Ende des ersten Studienabschnitts der Studienrichtung Angewandte Informatik stehen. Sie haben zu diesem Zeitpunkt nicht nur ausreichende Programmierpraxis und Datenbank-Kenntnisse, sondern auch relativ umfangreiche Kenntnisse über Software-Entwurf und Software-Entwicklungsmethodologie, sodaß ihnen die theoretischen Konzepte, auf denen die einzelnen Werkzeuge fußen, wohlbekannt und geläufig sind.

Weiters trägt diese Lehrveranstaltung viele Züge der Software-Entwicklung bei einem DV-Anwender oder in einem Software-Haus. Im Regelfall muß in Kleingruppen ein Projekt (eine Art "Gesellenstück") von der Analyse bis zu Test und Übergabe voll durchgezogen werden. Die Studierenden kommen somit nicht nur mit den Problemen der Aufgabenstellung sondern auch mit Problemen der Strukturierung einer Aufgabe in bezug auf zeitliche und personelle Aufgabenverteilung sowie Präsentation gegenüber Betreuer und Lehrveranstaltungsleiter in Berührung.

Im Wintersemester des abgelaufenen Studienjahres wurde - aufbauend auf Erfahrungen mit einem ähnlich gelagerten Vergleichsprojekt in bezug auf objektorientierte Programmierung - eine Strukturierung so gewählt, daß die Teilnehmer an dieser Lehrveranstaltung (13 Studenten) unter Koordination des Lehrveranstaltungsleiters in drei Dreipersonengruppen und eine Vierpersonengruppe gegliedert wurden, wobei jede Gruppe unter der Obhut eines eigenen Praktikumsleiters stand.

2.2. Evaluierte Software

Wie bereits in der Einleitung beschrieben, bezog sich das Experiment auf CASE-Tools für Forward-Software-Engineering (Analysis and Design Tools). Unter Berücksichtigung des verfügbaren Experimentier- bzw. Betreuungspersonals, der am Institut bereits vorab vorhandenen einschlägigen Produkte und der Marktsituation im deutschen Sprachraum wurde das Experiment mit den in Tab. 1 angeführten Werkzeugen und unter den dort angegebenen Software-Umgebungen durchgeführt.

Es wurden jeweils die zu Beginn des Experiments verfügbaren neuesten Produktversionen getestet. Lediglich bei dem von Ernst&Young vertriebenen CASE-Produkt wurde aus Verfügbarkeitsgründen nicht das neuere ADW sondern noch das vor einiger Zeit angeschaffte IEW getestet. Durch Kontakt mit ADW in anderen Kooperationsprojekten ist uns jedoch bewußt, daß ADW gegenüber IEW einige wesentliche Verbesserungen und Erweiterungen enthält und somit eine Extrapolation der Erfahrungen in Detailbereichen nicht möglich ist. Der Grundaufbau des Produktes ist aber im wesentlichen beibehalten geblieben.

IEW	Komponenten	Version 5.01 Planning Workstation (PWS) Analysis Workstation (AWS) Design Workstation (DWS) Construction Workstation (CWS)
	Umgebung	Windows 3.0 MS-DOS 5.0
Innovator	Komponenten	Version 4.1.1 Innovator Analysts Workbench (IAW:SERM, SA) Innovator Designers Workbench (IDW:SERM, SD) Innovator Programmers Workbench (IPW)
	Umgebung	Oracle Version 6.0.33 Open Look Version 3 Betriebssystem SunOS Version 4.1.1
Oracle*CASE	Komponenten	CASE*Designer 1.1.21 CASE*Dictionary 5.0.22 CASE*Generator 2.0.7
	Umgebung	SQL*Forms 3.0.16 SQL*ReportWriter 1.1.12 Oracle Version 6.0.33 Open Look Version 3 Betriebssystem SunOS Version 4.1.1
ProMod-PLUS	Komponenten	Version W 2.2 Information Modeling (IM) Stuctured Analysis (SA) Real Time Modeling (RT) Modular Design (MD)
	Umgebung	Open Look Version 3 Betriebssystem SunOS Version 4.1.1

Tab. 1: Die getesteten Tools und ihre Umgebungen

2.3. Experimentverlauf

Das "Praktikum aus Betriebsinformatik" ist grundsätzlich als einsemestriges Gruppenpraktikum vorgesehen, wobei der tatsächliche Aufwand pro Student zirka 500 Mannstunden beträgt. Jede Gruppe wird in regelmäßigen Gesprächen mit einem wissenschaftlichen Mitarbeiter des Instituts während der Dauer des Praktikums geführt.

In diesem Experiment wurden die Studierenden in einer Plenarveranstaltung aller Teilnehmer über Aufgabenstellung und Ziele des Projektes informiert und in die Thematik des zu bearbeitenden Anwendungsproblems eingeführt.

Als von den Studierenden zu lösende Aufgabe wurde die Erstellung eines Konferenz-Management-Systems gewählt. Diese Wahl wurde deshalb getroffen, weil in diesem Bereich seitens der Betreuer erhebliches KnowHow und seitens der Studenten grundsätzliches Vorverständnis vorlag. Darüberhinaus wurde dieser Anwendungskomplex bereits im Rahmen

der IFIP-CRIS-Studien breit publiziert, sodaß auch aus der Literatur eine Fülle von Musterlösungen, insbesondere aber eine klare Problembeschreibung vorlag. Weiters erlaubte diese Aufgabenstellung, die verwendeten Werkzeuge sowohl in bezug auf ihre Unterstützung von Datenbank-Problemstellungen als auch von prozeduralen Problemstellungen zu untersuchen.

Nach der Vorstellungsphase erhielten die Studierenden sowohl Dokumentation über das von der jeweiligen Gruppe zu verwendende CASE-Tool und die Aufgabenstellung, sich anhand von "Spielzeugbeispielen" und unter Anleitung des jeweiligen Praktikumsleiters mit dem Produkt vertraut zu machen. Darüberhinaus erhielten sie Verbalspezifikationen der Aufgabenstellung [OLLE 80], für die bereits in [OLLE 82] unterschiedliche Ausarbeitungen vorgelegt wurden, sowie die formale Spezifikation nach [OLLE 88].

Für die Einarbeitungsphase in das Werkzeug waren drei bis vier Wochen vorgesehen. Am Ende dieser Phase stand eine Plenarsitzung, in der die einzelnen Gruppen über das von ihnen verwendete Werkzeug berichteten und gleichzeitig ihr bis dahin gegebenes Problemverständnis über die Aufgabenstellung darlegen sollten.

Der weitere Projektverlauf sah vor, daß nach dieser Trainingsphase für das Werkzeug die eigentliche Arbeit an der Problemstellung beginnen sollte, wobei jeweils (im Ringtausch) ein Betreuer einer Nachbargruppe als "Kunde" fungierte, der in einem Initialinterview und in einer Reviewsitzung Wünsche und Korrekturwünsche formulierte (wobei diese zwischen den Betreuern weitestgehend abgestimmt waren). In einer dritten Plenarsitzung sollten die Studierenden den aufgrund dieser Interviews erarbeiteten Entwurf vorstellen, wobei geplant war, daß sich an diese Zwischenvorstellung des Entwurfs eine weitere Runde mit Änderungswünschen schließen sollte, um so auch die Unterstützung von Änderungserfordernissen durch einzelne Tools untersuchen zu können.

Tatsächlich fand zwar diese Zwischenrunde statt, es lag allerdings bis zu diesem Zeitpunkt bereits so viel an Änderungstätigkeit (Initialentwurf $\rightarrow$ 1. Interview-Runde $\rightarrow$ Review-Runde) vor, daß auf künstlich induzierte Änderungswünsche verzichtet werden konnte und nur jene Änderungswünsche eingebracht wurden, die zur vollständigen Abdeckung der Aufgabenstellung erforderlich waren (fairer Review). Die letzte Phase diente somit der Detailausarbeitung der vorgelegten Entwürfe und der weiteren Ausarbeitung, wobei diese aufgrund der unterschiedlichen Fähigkeiten der Tools teils in unterschiedliche Richtungen gingen.

Der Abschluß des Projektes verlagerte sich an den Beginn des Sommersemesters, um nicht durch Prüfungssituationen am Semesterende qualitative Einbußen in der Schlußphase des Experiments akzeptieren zu müssen. Er war durch eine Präsentation der ausgearbeiteten Lösung samt Dokumentation sowie einer Kurzbeschreibung des Werkzeuges (Zielgruppe: andere Studenten) und einen Erfahrungsbericht über das Werkzeug gegeben.

2.4. Validität des Experiments

Da es sich somit letztlich um Studierende handelte, die unterschiedliche CASE-Tools parallel auf eine identisch gelagerte Problemstellung anwandten, stellt sich die Frage, wie weit dieses

Projekt und seine Ergebnisse ein reales Abbild des praktischen Einsatzes von CASE-Tools in einem Software-Haus sein kann.

Wir sind der Meinung, daß sich dieses Projekt in mehrerer Hinsicht von "üblichen Studentenprojekten" unterscheidet.

1. Die Problemstellung hatte nichttrivialen Umfang und war auch inhaltlich so gewählt, daß sämtliche relevanten Modellierungsbereiche angesprochen und somit die eingesetzten Werkzeuge ziemlich vollständig ausgelotet wurden.

2. Die Studierenden hatten neben entsprechenden allgemein üblichen Informatik-lehrveranstaltungen vor Teilnahme an diesem Praktikum bereits eine umfassende (3+2 Semesterwochenstunden) Lehrveranstaltung aus Software-Engineering (Entwurf und Test) besucht.

3. Die Studierenden konnten in einer Einschulungsphase ihr theoretisches Wissen auf die Tool-Verwendung umlegen und waren somit zu Beginn der Bearbeitung der Aufgabenstellung sicherlich mindestens mit jenen Software-Entwicklern vergleichbar, die eine entsprechende mehrtägige Tool- und Methodenschulung beim Hersteller des CASE-Produkts besucht hatten und nun ihr erstes Projekt durchführten.

4. Durch die Praktikumsbetreuer stand stets eine hot-line zur Verfügung, wobei auch Kommunikation zwischen den Gruppen sicherstellte, daß die einzelne Gruppe nie mit ihrer Aufgabenstellung allein war.

5. Durch den Review-Plenar-Zwischentermin waren ausreichend viele Modifikationen (und de facto Neuentwicklungen) erforderlich, daß fairerweise angenommen werden darf, daß die Praktikumsteilnehmer bei Projektende mit jenen Praktikern vergleichbar waren, die bereits ihr zweites Projekt mit dem jeweiligen CASE-Tool hinter sich haben, und somit valide Aussagen, die von Anfangsfrust unbeeinflußt sind, treffen können.

Da dennoch in einem Projekt wie diesem Anfangsprobleme und persistente Probleme sich laufend überlagern und nur durch strikte Selbstbeobachtung und Bewußtseinmachung getrennt werden können, wollen wir die folgenden Ausführungen auch in eben diese beiden Gruppierungen gliedern.

3. ERFAHRUNGEN

Die in einem derartigen Projekt gewonnenen Erfahrungen sind kaum in vollständiger und systematischer Form darlegbar. Wir verzichten daher auch auf den Versuch einer derartigen Darstellung, sondern geben die Erfahrungen nach der subjektiven Gewichtung durch Betreuer und Studierende wieder. Dennoch erlaubt uns die Tiefe des Experiments eine Gliederung in solche (negativen) Erfahrungen, die grundsätzlich oder schwerpunktmäßig als Anfänger-probleme zu bezeichnen sind, in Installationsproblematiken und in jene Aspekte, die auch für den "Profi"-Benutzer als relevante Aspekte übrigbleiben.

3.1. Fallen für CASE-Anfänger

Zur korrekten Positionierung dieser Erfahrungen muß festgehalten werden, daß wir (bis auf eine Ausnahme, bei der eine detaillierte Produktdemonstration stattfand) keine formelle Herstellerschulung besuchten. Vielmehr hatten unsere Betreuer für Anfangsprobleme entsprechende Hilfestellung der Anbieter, mußten auf diese dank ihres Erfahrungsschatzes im Methodenbereich jedoch nur in Ausnahmefällen für gewisse Toolspezifika zurückgreifen.

Insbesondere die ersten drei der folgenden vier Punkte sind auf diesem Hintergrund zu verstehen. Dennoch halten wir unsere Situation nicht für zu abnorm, da insbesondere in größeren Unternehmen (bzw. bei entsprechender Personalfluktuation) sicherlich nicht davon auszugehen sein wird, daß jeder Verwender eines Tools sein diesbezügliches Wissen aus entsprechenden Herstellerkursen bezieht.

3.1.1. Tutorial

Ein Tutorial ist sicherlich auch nach entsprechenden Herstellerschulungen in die Verwendung eines CASE-Tools ein wesentliches Hilfsmittel zur Vertiefung des erworbenen Wissens. Wenn eine derartige Schulung nicht in Anspruch genommen wurde, ist seine Bedeutung für das Kennenlernen des Tools (sowohl in Papierform als auch in Form von Musterprojekten) umso stärker. Dies gilt auch für solche Projektmitarbeiter, die zwar in der verwendeten Methode geschult sind (wie unsere Studenten), jedoch die Handhabung des Tools an sich erst erlernen müssen.

Von den von uns untersuchten Tools stand uns nur für Oracle*CASE ein solches Handbuch zur Verfügung. Dieses Handbuch bezieht sich auf ein Musterbeispiel, das auch elektronisch verfügbar ist. Durch gleichzeitiges Arbeiten im Tutorial und am Bildschirm konnte schon nach kurzer Zeit ein guter Leistungsüberblick gewonnen werden.

Sowohl bei Innovator als auch bei ProMod-PLUS befanden sich Musterapplikationen im Lieferumfang des Tools. Im Falle von ProMod war diese Musterapplikation jedoch fehlerhaft, was gerade bei Anfängern zur Verwirrung beiträgt. Lobend muß aber erwähnt werden, daß die Studenten, die sich mit ProMod beschäftigten, direkt an der Universität eine eintägige Tooldemonstration durch den Vertreiber besuchen konnten. Für IEW stand eine elektronische Produktdemonstration zur Verfügung, die einen eher groben Überblick über die in den einzelnen Workstations angebotenen Methoden liefert.

3.1.2. Beschreibung der Tool-Philosophie

Da CASE-Tools üblicherweise aus mehreren Komponenten bestehen, ist es notwendig, daß der Anwender mit dem Zusammenspiel dieser Komponenten vertraut gemacht wird, um das Tool auch in richtiger Art und Weise einsetzen zu können.

Keine der vier Tooldokumentationen weist explizit eine derartige Beschreibung der Tool-Philosophie auf. Am ehesten bieten noch Innovator und Oracle*CASE jeweils sehr kurze diesbezügliche Informationen. Bei Oracle*CASE beschränkt sich diese auf ein paar Diagramme ohne erläuternden Text und ist eher als Nachschlagehilfe denn als Einstiegshilfe verwendbar.

3.1.3. Verwendete Notation
Die Erfahrungen der Studenten aus Entwurfslehrveranstaltungen stützen sich vorwiegend auf Standard-Notationen, insbesondere [CHEN 76] bezüglich ER-Modellierung und [YOUR 79] hinsichtlich SA/SD. Sie sind damit in jenen Methoden und Notationen ausgebildet, die im universitären Bereich als Quasi-Standard gelten können.

ProMod verwendet im ER-Bereich die Standard-Notation nach [CHEN 76]. IEW und Oracle*CASE verwenden demgegenüber die davon leicht abweichende Notation von Gane/ Sarson [GANE 79]. Die größten notationellen Abweichungen traten bei Innovator auf, da dieses Werkzeug die in [SINZ 87] vorgeschlagene strukturierte ER-Notation (SERM) verwendet. Dementsprechend traten hier auch die größten Einarbeitungsschwierigkeiten auf, da anfänglich die Methode falsch angewandt wurde und erst nach Beschaffung von Sinz's Habilitationsschrift [SINZ 87] der Großteil der Unklarheiten ausgeräumt war.

Hinsichtlich der strukturierten Analyse unterstützen Innovator und ProMod zusätzlich zur de-Marco-Methode [DEMA 78] den Entwurf von Echtzeitsystemen (vgl. [WARD 85]). Oracle*CASE und IEW verwenden weiterhin Gane/Sarson bzw. leichte Abänderungen davon.

Für den strukturierten Entwurf (SD) verwenden IEW, Innovator und ProMod die Standardmethode nach Yourdon/Constantine [YOUR 79]. Oracle*CASE sieht kein SD vor.

3.1.4. Referenzhandbuch
Ein Referenzhandbuch beinhaltet Informationen, die sowohl für den Anfänger unabdingbar, jedoch auch für den CASE-Experten insbesondere zum Überwinden schwieriger (oder seltener) Klippen erforderlich sind.

Die im allgemeinen gewählte Strukturierung nach Menüpunkten mag zwar als Nachschlaghilfe für Vertiefungsfragen adäquat sein, bedingt jedoch sowohl bei Anfängern als auch in jenen Situationen, in denen man nach entsprechender Erfahrung im Tool fachliches Neuland betritt, erhöhten Suchaufwand.

Für alle von uns betrachteten Tools sind derartige Referenzhandbücher vorhanden (unabhängig von der unter 3.1.1. bzw. 3.1.2. erwünschten, jedoch meist nicht verfügbaren Einstiegsliteratur). Am Beispiel SERM von Innovator zeigte sich jedoch, daß bei Unkenntnis einer Methode ein Referenzhandbuch auch nichts nützt. Für den der toolspezifischen Fachausdrücke Unkundigen haben sich auch im Falle von IEW Probleme beim Rückgriff auf das Referenzhandbuch bemerkbar gemacht. Für den methoden- und toolvertrauten Anwender sind die mitgelieferten Referenzhandbücher jedoch durchwegs als ausreichend empfunden worden.

3.2. Installationsprobleme

3.2.1. Installation des Tools
Bis auf Oracle*CASE, dessen Installation teilweise von Oracle selbst durchgeführt wurde, wurden alle anderen Tools von institutseigenen Technikern installiert. Bei Installation der Tools selbst sind keine wesentlichen Probleme aufgetaucht. Es gab jedoch einige zeitaufwendige Detailprobleme bezüglich der Peripherie.

Diese Schwierigkeiten bezogen sich hauptsächlich auf die Druckerausgabe von Reports und Graphiken (siehe 3.3.7.). Zusätzlich führte bei Oracle*CASE die Initialisierung der Funktionstasten zu Problemen. So konnten in den meisten Menüs häufig verwendete Funktionen nur über work-arounds (meistens zwei bis drei Tasten pro Funktion) angesprochen werden, was den Arbeitsfluß deutlich verminderte.

3.2.2. Handhabung des Servers
Innovator, Oracle*CASE und ProMod-PLUS verfügen über eine Client-Server-Architektur. Anfängliche Probleme im Umgang mit dem Server traten bei den Gruppen auf, die sich mit Innovator und ProMod beschäftigten, da die zugrundeliegende Philosophie ungenügend beschrieben war. Beispielsweise suchten die Mitglieder beider Gruppen lange Zeit vergeblich nach einer expliziten COMMIT-Möglichkeit. Für zusätzliche Verwirrung sorgte im Falle von Innovator die Notwendigkeit, den Server projektspezifisch anzusprechen.

3.3. "Kleinigkeiten", die auch den Profi ärgern

In diesem Abschnitt wollen wir auf Erfahrungsaspekte eingehen, die von der spezifischen Position in der Lernkurve bezüglich des Tool-Einsatzes unabhängig sind. Wir behandeln diese zuerst nach Sachgebieten, die für alle untersuchten Tools in gleicher Weise relevant sind, und konzentrieren uns dann auf Aspekte, die nur bei einzelnen Tools auftraten oder nur bei diesen relevant sind. Die Reihung der Nennung entspricht wieder stärker dem subjektiven Eindruck denn irgendeiner Systematik, sodaß etwa die scheinbare Kleinigkeit eines "UNDO" unmittelbar vor dem Makro-Aspekt des Konfigurationsmanagements diskutiert wird.

3.3.1. Zusammenhang zwischen den einzelnen Tool-Komponenten
Sinnvollerweise sollte ein CASE-Tool alle Phasen des Software Life Cycles unterstützen. Dies bedeutet jedoch, daß verschiedene Methoden angeboten werden müssen. Nun stellt sich die Frage, inwieweit die einzelnen Methoden integrierbar sind, bzw. welche Ergebnisse wie gut aus anderen Methoden bzw. Diagrammen übernommen werden können.

Alle Tools verfügen über ein Data-Dictionary, auf das die einzelnen Tool-Komponenten zugreifen. Nur SD in ProMod bildet hier einen Sonderfall und greift nicht auf die Definitionen im Data-Dictionary zu. Falls Änderungen jedoch direkt im Data-Dictionary gemacht werden, werden diese bei IEW und Innovator sofort in den jeweiligen Diagrammen automatisch nachvollzogen; Oracle*CASE benötigt dazu eine eigene Benutzereingabe; ProMod bietet keine automatische Konsolidierung.

Innovator bietet durchwegs Anknüpfungspunkte zwischen den einzelnen Methoden, was zum Beispiel zwischen SERM-Diagramm und SA durch Views, die Datastores zugewiesen werden können, besonders gut gelöst ist. In Oracle*CASE und IEW können Daten aus dem ER-Diagramm über das Dictionary bequem in das Datenflußdiagramm übernommen werden. ProMod sieht zwar die Übernahme von bereits definierten Entities und Assoziationen vor, die damit verbundenen Aktivitäten werden jedoch nur mäßig unterstützt.

Werden in IEW, Innovator und Oracle*CASE Entities und Attribute, die in einem Datenflußdiagramm verwendet werden, im ER-Diagramm gelöscht, so gibt es automatisch Auswirkungen auf die entsprechenden Datenflüsse und Datastores.

3.3.2. Konsistenzüberprüfungen
Von allen getesteten Tools werden einzelne Menüpunkte angeboten, die unterschiedliche Konsistenzchecks innerhalb einzelner Methoden durchführen, wobei diese Checks von Tool zu Tool qualitativ variieren. Innovator bietet zusätzlich die Möglichkeit die Überprüfungsbedingungen benutzerspezifisch zu parametrisieren.

3.3.3. Explizites COMMIT bzw. UNDO
Bei der interaktiven Entwurfserstellung ist es wesentlich, (vor kurzem) getroffene Entscheidungen zumindest in bescheidenem Maße wieder rückgängig machen zu können bzw. auf ältere und vielleicht bessere Versionen zurückgreifen zu können.

Durch die Datenbank-Erfahrung des Herstellers hat man im Falle von Oracle*CASE wohl erkannt, daß ein Mindestausmaß eines Transaktionskonzeptes erforderlich ist; es gibt hier die Möglichkeit eines Rollback bis zur letzten bestätigten Datenbankänderung. ProMod stellt während des Arbeitens sogar eine schrittweise UNDO-Funktion zur Verfügung, jedoch nur in den einzelnen Graphikeditoren. Bei IEW und Innovator sind weder ein explizites COMMIT noch eine UNDO-Funktion vorgesehen, was natürlich ein sehr konzentriertes Vorgehen bei der Arbeit mit dem Tool erfordert.

3.3.4. Konfigurationsmanagement
Um Versionen und Varianten eines Projekts verwalten zu können, ist ein Konfigurationsmanagement insbesondere bei Teamarbeit bzw. der Verfolgung von Produktvarianten notwendig. Wenn man bedenkt, daß einige Tools kein explizites COMMIT bzw. UNDO unterstützen, wäre ein geeignetes Konfigurationsmanagement hier sicherlich eine (zwar unschöne, aber immerhin) Abhilfe. Bis auf Oracle*CASE, das eine Versionsverwaltung vorsieht, bieten die untersuchten Tools jedoch nichts dergleichen an.

3.3.5. Benutzeroberfläche
Alle getesteten Tools bieten eine graphische Benutzeroberfläche an. Innovator und IEW bieten eine einheitliche Benutzerschnittstelle für alle Tool-Komponenten inkl. einer On-line-Hilfe, die im wesentlichen aus dem Referenzhandbuch besteht; es werden fast überall Auswahllisten, bestehend aus den bisherigen Einträgen im Data-Dictionary, angeboten, um bestehende Objekte leichter anzusprechen und wiederzufinden. Bei Innovator ist allerdings negativ aufgefallen, daß man bei der Suche nach einen bestimmten Datenfluß im Bubble-Chart keine automatische Unterstützung erhält. Im Falle von IEW ist zu bemängeln, daß der Wechsel zwischen den einzelnen Workstations relativ aufwendig ist.

Bei ProMod ist anzumerken, daß sich die Dokumentation auf die Oberfläche einer anderen Hardware-Plattform bezog. Es bestanden darüber hinaus keine direkten Übernahmemöglichkeiten aus dem Data-Dictionary. Oracle*CASE hat zwei Benutzerschnittstellen mit völlig unterschiedlicher Bedienung. So gibt es einerseits die Oberfläche für die einzelnen Graphikkomponenten, in denen Diagramme gezeichnet werden, und eine zeichenorientierte SQL*Forms-Oberfläche, über die alle restlichen Entwurfsdaten eingegeben werden. Das ständige Umschalten zwischen der mausorientierten und der tastenorientierten Oberfläche wurde als gewöhnungsbedürftig empfunden.

3.3.6. Defaultdarstellungen
Um schnell ein lesbares Diagramm zu erzeugen, sind Defaultdarstellungen und ein vernünftiges Auto-Routing von Vorteil.

Oracle*CASE und ProMod bieten keine Defaultdarstellungen; die restlichen zwei Tools bieten zwar Defaultdarstellungen, diese sind jedoch bei größeren Diagrammen durchwegs unübersichtlich. Außerdem ist vor der Verwendung von Default-Repräsentationen bei IEW und Innovator zu warnen, wenn man bedenkt, daß hier keine UNDO-Funktion mehr möglich ist.

3.3.7. Reportgenerierung
Die Generierung von Reports führte im Falle von Innovator, Oracle*CASE und ProMod zu erheblichem Speicherbedarf, weil dabei Druckfiles erzeugt wurden, die im zwei- bis dreistelligen Megabyte-Bereich lagen. ProMod erzeugte außerdem ziemlich unübersichtliche Reports, insbesonders komplexe Diagramme konnten nicht mehr entziffert werden, da diese genau in ein A4-Blatt gepreßt werden. Um schließlich Graphiken einzeln (Innovator) und lesbar (IEW, Oracle*CASE) ausdrucken zu können, wurden als Behelfslösung Snapshots vom Bildschirm erzeugt.

3.4. Die Tools im einzelnen

3.4.1. Die getesteten Tools auf einen Blick

	IEW	**Innovator**	**Oracle*CASE**	**ProMod-PLUS**
ER-Modellierung	ja	ja	ja	ja
Notation	Gane/Sarson	Sinz (SERM)	Gane/Sarson	Chen
mehrstell. Beziehungen	nein	ja	nein	ja
Beziehungsattribute	nein	ja	nein	ja
Aggregation	nein	ja	ja	nein
Generalisation	nein	ja	nein	ja
NF-Generierung	1. NF	3. NF	nein	nein
Datendefinition	hier. + rel. Mod.	relational	relational	nein
Views	ja	NF2-Views	ja	nein
Anbindung ER -> SA	ja	ja	ja	mäßig
Structured Analysis	ja	ja	ja	ja
Notation	Gane/Sarson	Yourd./Const.	Gane/Sarson	Yourd./Const.
Daten- u. Kontrollflüsse	ja	ja	ja	ja
Prozeßhierarchie	ja	ja	ja	nein
Disjunktion/Konjunktion	nein	nein	nein	nein
Realtime-Erweiterung	nein	ja	nein	ja
Anbindung SA -> SD	mäßig	ja	kein SD	nein
Structured Design	ja	ja	nein	ja
Notation	Yourd./Const.	Yourd./Const.		Yourd./Const.
Datenspeicher	nein	ja		nein
Kontrollstrukturen	nein	nein		nein
Daten- u. Kontrollflüsse	ja	ja		nur Datenflüsse
Structured Progr.	ja	ja	nein	ja, nicht getestet
Notation	Modul Action D.	Nassi-Shneiderman		Texteditor
Bildschirmmasken	ja	Zusatz-Tool JAM	SQL*Forms	nein
Code-Generierung	ja, nicht getestet	ja	ja	ja, nicht getestet
Sprachen	Cobol, SQL	C, Pascal, Fortran PL/M, SQL	SQL-Forms, SQL-Menu	C, Fortran
Zielrichtung	3GL, Datenbank	3GL, Datenbank	4GL	3GL, 4GL

Tab. 2: Die Tools im Vergleich

3.4.2. IEW

ER (PWS, AWS):
IEW verwendet bei den ER-Diagrammen eine Notation, die der von Gane/Sarson sehr ähnlich ist. Die Kardinalitäten der Beziehungen werden jedoch in Ergänzung dazu durch Intervalle beschrieben.

Gravierende Unterschiede zu den klassischen Ansätzen der ER-Modellierung liegen darin, daß sich einerseits keine mehrstelligen Beziehungen darstellen lassen und andererseits Beziehungen keine Attribute aufweisen können. Um die Semantik mehrstelliger Beziehungen darstellen zu können, müssen diese unter Verwendung "assoziativer Entities" in binäre Beziehungen umgewandelt werden. Diese assoziativen Entities, die lobenswerterweise eine eigene Darstellungsform besitzen, werden auch zur Definition von Beziehungsattributen herangezogen.

Als unnötig aufwendig wurde die Definition von Schlüsselkandidaten, die sich aus mehreren Attributen zusammensetzten, empfunden. Es müssen nämlich alle Attribute, die den Schlüsselkandidaten bilden sollen, zu einem "Meta-Attribut" zusammengefaßt werden, welches dann durch entsprechende Festsetzung der Kardinalitätsgrenzen und seines maximalen Vorkommens als Schlüsselkandidat zu kennzeichnen ist. Was nicht überprüft wurde, da die entsprechende Situation in unserem Projekt nicht auftrat, ist die Reaktion des Tools bei der Transformation in eine Datenbasis, wenn es mehr als einen Schlüsselkandidaten gibt.

Vermißt werden modernere Konzepte wie Aggregation und Generalisation. Letztere ist in ADW durch Einführung von Multityping-Konzepten (super- und subtypes) realisiert worden.

Weiters sei hier noch erwähnt, daß dieses CASE-Tool das einzige war, welches wir auf einer PC-Plattform getestet haben. Die dabei gewonnene Erfahrung zeigt auf, daß die bei PCs übliche Bildschirmgröße für ein effektives Arbeiten bei weitem nicht ausreicht. So erschöpft sich die Lesbarkeit bzw. Überblickbarkeit der Diagramme schon bei mäßiger Komplexität.

SA (AWS):
SA-Dokumente können mit IEW auch unabhängig von der konzeptuellen Modellierung verwendet werden. Anbindungsmöglichkeiten existieren sowohl vom Dataflow-Diagramm zum ER-Diagramm als auch umgekehrt.

Bei den Datenflußdiagrammen wird eine zu Gane/Sarson ähnliche, aber erweiterte Notation verwendet. Im Gegensatz zu den klassischen Methoden ist die Darstellung von Konjunktion und Disjunktion der Datenflüsse nicht möglich. Außerdem sind Prozesse, für die es Verfeinerungsdiagramme gibt, im DFD nicht als solche gekennzeichnet. Ob eine Verfeinerung existiert, kann man nur durch Anwahl eines eigenen Menüpunktes oder durch Betrachtung des "Decomposition Diagram" feststellen. Als positiv ist hingegen anzusehen, daß das Tool die Konsistenz der Anzahl der Input- und Outputströme über alle Verfeinerungsstufen prüft.

SD/SP (DWS):
SD-Entwürfe können ebenfalls ohne vorangegangene Analyseschritte (ER und SA) erstellt werden. Es gibt leider keine Möglichkeit, Structure Charts aus den in der AWS erstellten DFD (halb)automatisch ableiten zu lassen.

Die bei den Sturcture Charts verwendete Notation entspricht der von Yourdon/Constantine, wobei es jedoch nicht möglich ist, Datenspeicher und Konstrollstrukturen darzustellen.

Als unnötig hoch wurde der Aufwand empfunden, der sich durch bestimmte Modifikationen in den Moduldefinitionen ergab. So können zwar die Namen der formalen Parameter eines Moduls leicht geändert werden, will man aber aus einem Input-Parameter einen Output-Parameter (bzw. vice versa) machen, muß das gesamte Modul gelöscht und wieder, entsprechend modifiziert, neu angelegt werden.

Für den detaillierten Design und als notwendige Voraussetzung für die spätere Code-Generierung sind für alle Module des Structure Charts sogenannte "Module Action Diagrams" zu erstellen. Dazu können die bereits in der AWS erstellten MiniSpecs (= Action Diagrams), die unter Verwendung eines Pseudocodes die Aktionen (Abfragen, Datenzugriffe etc.) der im DFD dargestellten Prozesse blockdiagramm-ähnlich beschreiben, als Grundlage herangezogen und entsprechend erweitert werden.

Problematisch ist der Umstand zu nennen, daß das Tool keine Möglichkeit anbietet, Daten und Objekte direkt aus dem Datenflußdiagramm ins Module Action Diagram zu übernehmen. Da vieles neu angelegt werden muß, sind Fehler und Inkonsistenzen zwischen den beiden Diagrammen sehr wahrscheinlich.

Weiters soll bemerkt werden, daß sich die Module Action Diagrams sehr nahe am Code-Niveau bewegen. Die dabei verwendete Notation lehnt sich stark an COBOL an, sodaß Grundkenntnisse in dieser Programmiersprache für die Arbeit erforderlich sind.

Generator:
Um die Codegenerierung durchführen zu können, ist es unter anderem notwendig, aus dem modellierten ER-Diagramm ein Datenbankdesign abzuleiten (es sind auch alle Action Diagramme zu erstellen). Die Erstellung der Database stellt einen wesentlichen Schnitt in der Entwicklungsphase dar, da Änderungen im Design, die danach durchgeführt werden, keine Auswirkungen mehr auf das ER-Diagramm haben und umgekehrt Änderungen im ER-Diagramm nicht mehr bis in die Database nachgezogen werden - außer man generiert sie wieder neu.

ADW bietet hier den Vorteil, daß nach Generierung der ersten Database nicht eine vollständig neue Transformation durchgeführt werden muß, sondern nur die geänderten Teile iterativ nachgezogen werden können.

Anmerkung betreffend aller IEW-Toolkomponenten:
Aus naheliegenden Gründen wird während des Projektverlaufes unterschiedlich häufig ein Komponentenwechsel erforderlich sein. Weil jede Komponente von IEW eine eigene Paßwortabfrage enthält, wurde dieser Wechsel als besonders mühsam empfunden. Begründet wurden diese Abfragen vom Hersteller damit, daß jede Toolkomponente auch einzeln einsetzbar ist.

3.4.3. Innovator

ER:
Wie bereits unter 3.1.3. erwähnt wurde, unterscheidet sich die von Innovator umgesetzte Entity-Relationship-Modellierung nicht unwesentlich vom klassischen Ansatz. Dieser Unterschied wurde anfangs nicht erkannt und führte somit zu Fehlinterpretationen der verschiedenen angebotenen Konstrukte. So wurden z.B. Kardinalitäten als Konnektivitäten aufgefaßt, was zu nicht darstellbaren zirkulären Beziehungen im ER-Diagramm und zu Objekten führte, die ein und denselben Fremdschlüssel mehrmals beinhalteten; außerdem wurde der R-Typ fälschlicherweise als mit n:m-Beziehungen gleichwertig betrachtet; überhaupt war der Unterschied zwischen ER- und R-Typ ziemlich unklar; zusätzliche Verwirrung stiftete die Links-Rechts-Hierarchie von Objekten als Konsequenz der Existenzabhängigkeit (das herkömmliche ER-Modell hat keine Einschränkungen bezüglich Objektpositionierung). Zu diesem Zeitpunkt wurde also klar, daß doch gravierende Unterschiede zur herkömmlichen ER-Modellierung bestehen mußten. Erst nach Beschaffung und Durchsicht der Originalliteratur ([SINZ 87]) zur strukturierten ER-Modellierung (SERM) erkannte man das Ausmaß der falschen Fährte, die verfolgt wurde - es erwies sich als das Beste, die bisherigen Entwurfsdokumente zu verwerfen und die Datenmodellierung des Einstiegsbeispiels neu zu beginnen.

Abgesehen davon, daß das Arbeiten mit dieser neu erlernten Methode etwas gewöhnungsbedürftig war, haben sich im weiteren Verlauf des Projekts überhaupt keine Probleme mehr mit der SERM-Analysts- und später mit der SERM-Designers-Workbench ergeben. Um das eher komplexe SER-Diagramm übersichtlicher zu gestalten, wurde auch von der Möglichkeit der Aggregationsbildung Gebrauch gemacht; die diesbezüglichen hierarchischen Zusammenhänge können zusätzlich in Form eines Objektbaums repräsentiert werden.

Erwähnenswert ist die Verbindung des SER-Diagramms mit der strukturierten Analyse durch die Bildung von Views, die Stores im Bubble-Chart zugeordnet werden können. Zur Wartung dieser Views steht der sogenannte Projektbaum zur Verfügung. Dieser ist das einzige Instrument, um Views auch wieder vollständig aus dem Projekt entfernen zu können.

SA:
SA-Entwürfe können mit Innovator auch unabhängig von der konzeptuellen Modellierung erstellt werden. Wegen der angebotenen Anbindungsunterstützung zur Datenmodellierung wird aber empfohlen, mit SERM zu beginnen und die dort definierten Objekte in der strukturierten Analyse weiterzuverwenden. Abgesehen von der Real-Time-Erweiterung in Form von State-Transition-Diagrammen bestehen zur Notation nach [Your79] nur zwei Abweichungen: Zum einen können Disjunktion und Konjunktion von Datenflüssen im Datenflußdiagramm nicht dargestellt werden, zum anderen werden Datenflüsse bereits am SA-Niveau mit Datentypen gleichgesetzt. Letztere Tatsache behindert ein lehrbuchmäßiges Vorgehen, bei dem inhaltliche Veränderungen, die Datenflüsse von Prozeß zu Prozeß erfahren, auch in unterschiedlichen Datenflußbezeichnungen ihren Niederschlag finden sollten.

Positiv (wenn auch nur zufällig) ist hingegen aufgefallen, daß die Anzahl der möglichen Prozesse pro Repräsentationsebene, die Art der Prozeßnumerierung und einige weitere nützliche Einstellungen durch Editieren einer Parameterdatei direkt vom Tool-Benutzer bestimmt werden können. Außerdem wird in Form eines Prozeßbaumes eine übersichtliche Darstellung der Prozeßhierarchie des Gesamtprojekts geboten. Schließlich sind Bubbles, zu

denen eine Verfeinerungsebene existiert, bereits im Datenflußdiagramm besonders gekennzeichnet und somit sofort erkennbar.

SD:
SD-Entwürfe können ebenfalls ohne vorangegangene Analyseschritte mittels SERM und SA erstellt werden, wobei jedoch Einschränkungen hinsichtlich der verwendbaren Parametertypen bestehen, die entweder in der Datenelementverwaltung der SERM-Analysts-Workbench definiert worden sein müssen oder den bereits vom Tool vordefinierten C-Basistypen entsprechen müssen.

Bei Anknüpfung an bestehende SA-Diagramme werden alle während der strukturierten Analyse deklarierten Daten- und Kontrollflüsse für den strukturierten Design als Typen zur Verfügung gestellt. Außerdem wird die Zuweisung von Prozessen, Stores und Terminatoren aus SA-Diagrammen an Operationen in SD-Diagrammen unterstützt, wobei auch die ursprünglichen Aufrufhierarchiestufen aus SA übernommen werden.

Im Zusammenhang mit SD ist aufgefallen, daß Kontrollstrukturen nicht dargestellt werden können und daß in weiterer Folge somit nur die Aufrufschnittstellen als Prozedurköpfe an den Nassi-Shneiderman-Editor übergeben werden. Kontrollstrukturen scheinen somit erst im Nassi-Shneiderman-Editor der Programmers Workbench auf.

SP:
Für die strukturierte Programmierung wird durch die Programmers Workbench ein Nassi-Shneiderman-Editor zur Verfügung gestellt. Der Nassi-Shneiderman-Editor unterstützt die Top-Down-Vorgangsweise, da durch sogenannte Folders Probleme in Teilprobleme zerlegt werden können. Mit Hilfe einer spezifischen Parameterdatei können Bezeichnungen von Kontrollstrukturen in Struktogrammen an Benutzeranforderungen angepaßt werden. Die Programmiersprache, die für die Erstellung von Struktogrammen herangezogen werden soll, kann mit dem SA/SD-Projektserver angewählt werden. Es wird empfohlen, die Sprache so spät wie möglich zu fixieren, da sich herausgestellt hat, daß eine einmal eingestellte Sprache für das gesamte Projekt unwiderrufbar ist.

Hinsichtlich der Stabilität des Tools erwies sich der Nassi-Shneiderman-Editor im Falle des Aufrufs innerhalb der integrierten Innovator-Umgebung als die absturzgefährdetste Komponente. Die aus Views generierbaren Eingabemasken konnten nicht erzeugt werden, da dazu der Maskengenerator JAM notwendig wäre, der jedoch nicht gemeinsam mit der Innovator CASE-Workbench vertrieben wird. Somit konnte die Codegenerierung nur teilweise getestet werden.

3.4.4. Oracle*CASE

ER:
Wie in IEW lassen sich auch in diesem CASE-Tool keine mehrstelligen Beziehungen direkt darstellen. Werden solche gebraucht, müssen dazu Entities verwendet werden, die nur Beziehungssemantik haben. Eine besondere Kennzeichnung dieser "Beziehungs-Entities" ist nicht möglich, was bei komplexeren ER-Diagrammen die Lesbarkeit herabsetzt. Zusätzlich unterstützt das Tool keine Attribute bei Beziehungen. Sollen Beziehungen Attribute haben, müssen wieder "Beziehungs-Entities" verwendet werden.

Weiters beschränkt sich die Kardinalitätenvergabe bei Beziehungen auf die klassischen drei Typen (1:1, 1:n, n:m) und hält damit nicht mit den neueren Vorschlägen (wie z.B. Ober- und Untergrenzen) Schritt.

SA:
Als gut gelöst wurde der Übergang vom "Function-Hierarchier" und dem Datenflußtool empfunden. So können im "Hierarchier" die Funktionen definiert werden, und diese Funktionen können bequem im Datenflußtool weiterverarbeitet werden. Modifikationen in der einen Toolkomponente wirken sich entsprechend auf die andere Komponente aus. Allerdings ist ein Schließen und Öffnen der anderen Komponente nötig, um diese Änderungen sichtbar zu machen.

Funktionen werden auf der Generatorseite als Module implementiert. Wurde für eine Funktion schon ein Modul definiert, kann diese im "Hierarchier" nicht mehr gelöscht werden. Erst nachdem das zughörige Modul gelöscht wurde, kann auch die Funktion entfernt werden.

Wurde eine Funktion im "Hierarchier" gelöscht, scheint sie im Datenflußdiagramm noch weiter auf. Erst ein Löschen der Funktion über einen entsprechenden Menüpunkt im "Datenflußdiagrammer" beseitigt sie auch hier. Pikanterweise wird man hierbei vom System gefragt, ob auch ein Löschen im Data-Dictionary erwünscht sei. Diese Frage muß der Benutzer mit "Nein" beantworten, da es sonst zu einer Systemfehlermeldung kommt, da die Funktion dort schon durch den Löschvorgang im "Hierarchier" gelöscht wurde.

Erschwert wird das Arbeiten durch den Umstand, daß Funktionen, für die es Verfeinerungen gibt, nicht im "Datenflußdiagrammer" gekennzeichnet sind. Entweder "Trial and Error" oder gleichzeitiges Offenhalten von "Hierarchier" und "Datenflußdiagrammer" sind praktikable work arounds. Nicht unterstützt werden Konjunktion und Disjunktion von Datenflüssen.

Generator
Nach Erstellung des ER-Diagrammes kann man über dem Menüpunkt "default database design" die Datendefinitionen generieren. Allerdings kommt es dabei zu keiner Löschung der bereits generierten Definitionen. Wurde beispielsweise im Entwurf zwischen zwei aufeinanderfolgenden Durchläufen ein Entity gelöscht, existiert die dem Entity entsprechende Definition weiter in der Datenbank und muß händisch gelöscht werden.

3.4.5. ProMod-PLUS

ER:
Die "Information Modeling"-Komponente wurde von den ProMod-Testern als mit Abstand bester Teilbereich dieses Tools bezeichnet. Dies wohl auch nicht zuletzt deshalb, weil sich ProMod strikt an die wohlbekannte Quasi-Standard-ER-Notation nach [CHEN 79] hält. In Ergänzung zur Erstellung von ER-Diagrammen bietet ProMod die Möglichkeit der Zusammenfassung von einzelnen Diagrammen zu sogenannten ER-Modellen, die jedoch selbst nicht mehr erweiterbar sind. Die Diagrammaufteilung hingegen wird von ProMod nicht unterstützt.

Nur für die erstmalige Repräsentation auf ER-Modell-Ebene wird eine Defaultdarstellung angeboten, die jedoch unbefriedigend ist, da es dem Benutzer obliegt, die einzelnen (meist

überlagerten) Diagrammteile selbst graphisch zu entflechten. Zusätzlich traten Probleme nur bei direkt im Data-Dictionary vorgenommenen Löschungen und Änderungen von Datendefinitionen auf, die im ER-Diagramm nicht automatisch nachvollzogen wurden.

SA:
Bei der SA-Komponente von ProMod fiel die unzureichende Unterstützung der Übernahme von Ergebnissen aus dem ER-Diagramm besonders auf. Außerdem finden Namensänderungen von Prozessen, Datenflüssen und Stores keine automatische Unterstützung. Disjunktion und Konjunktion von Datenflüssen können nicht dargestellt werden. Terminatoren können nur im Kontextdiagramm - also auf höchster Hierarchiestufe - verwendet werden. Außerdem darf das Kontextdiagramm nur aus einem einzigen Prozeß bestehen und keine Stores beinhalten. Werden diese Vorschriften bei der Modellierung nicht eingehalten, erfolgen entsprechende Warnungen durch den sogenannten "Analyzer". Dieser Analyzer verlangt zwar, daß in einem Datenflußdiagramm ein Prozeß die Nummer 0 zu tragen hat, die Vergabe von Default-Nummern beginnt jedoch bei 1, was bedeutet, daß entweder alle Nummern händisch ausgebessert oder die Fehlermeldungen des Analyzers zwangsläufig ignoriert werden müssen. Änderungen von Datenflußbezeichnungen müssen in allen Diagrammen händisch nachvollzogen werden. Weiters sind verfeinerte Prozesse im Diagramm nicht eigens gekennzeichnet. Zur Dokumentation von Prozessen können MiniSpecs herangezogen werden.

SD:
Ein besonderer Kritikpunkt an der SD-Komponente von ProMod besteht wohl darin, daß keine Unterstützung bei der Übernahme von Ergebnissen aus der strukturierten Analyse existiert. Diese Unabhängigkeit zu den anderen Tools hat zumindest den Vorteil, daß bei Änderungsoperationen eher wenig Konsistenzverletzungen auftreten können. Auch hier gilt, daß bereits verfeinerte Elemente nicht speziell gekennzeichnet sind. Weiters fiel auf, daß in Stucture Charts keine Kontrollflüsse, Datenspeicher und Kontrollstrukturen darstellbar sind.

4. PROJEKT-ERGEBNISSE

4.1. Projektumfang - Was nicht getestet wurde

Zum Projektumfang kann abschließend festgestellt werden, daß hier eine vergleichsweise komplexe Applikation analysiert, dokumentiert und entworfen wurde. Als Indikatoren für den Projektumfang mögen folgende Eckdaten dienen: Im Schnitt wurden ca. 20 Entitäten definiert und 30 Bildschirmmasken entworfen. Die SA-Diagramme erstreckten sich auf vier bis sechs Hierarchiestufen. Der Gesamtumfang der abgelieferten Dokumentation variierte stärker zwischen den Gruppen, wies allerdings in allen Fällen eine dreistellige Seitenanzahl auf.

Die Entity-Relationship Modellierung sowie Structured Analysis und der darauf aufbauende Entwurf wurden von allen Gruppen durchgeführt. Die daran anschließenden Tätigkeiten wichen aufgrund der Spezifika der einzelnen Tools jedoch leicht ab bzw. wurden nur mehr punktuell ausgeführt. Dies gilt insbesondere für die Code-Generierung. Bei Innovator wurden C-Libraries und Oracle-Schnittstellen generiert, aus Oracle*CASE wurden Masken für eine Prototyp-Applikation in SQL*Forms generiert. Bei ProMod hatten wir die entsprechende Werkzeug-Komponente (Source Pilot) nicht zur Verfügung, bei IEW wurde die Code-Generierung aufgrund von Kompatibilitätsproblemen zu den verfügbaren COBOL-Compilern nicht durchgeführt.

Weiters wurden die von einzelnen Anbietern angebotenen Reverse-Engineering- und Prototyping-Komponenten nicht in die Untersuchung einbezogen.

4.2. Was wir uns von CASE-Anbietern wünschen würden

4.2.1. Handbücher

Obzwar von allen Anbietern entsprechende Handbücher vorlagen, erwiesen sich diese generell für ein Publikum, das keine Herstellerschulung besuchen konnte, als wenig geeignet.

Vor allem wurde das Fehlen von Musterlösungen und das Fehlen von Toolbeschreibungen auf hohem Niveau (insbesondere Tool-Zusammenhangsdokumentation) bemängelt. Letztere existieren zwar am Niveau von Werbebroschüren, nicht jedoch in einem für den Benutzer hilfreichen Detaillierungs- und Präzisionsgrad. Insbesondere Benutzern mit entsprechender Methodenkenntnis oder Kenntnis eines Konkurrenzproduktes (aus ihrer Arbeit vor einem allfälligen Jobwechsel) wäre damit der Einstieg in ein Werkzeug sehr erleichtert. In gleicher Weise würden solche Unterlagen für die Tool-Auswahl vor Erstanschaffung hilfreich sein.

4.2.2. Durchgängigkeit

Hier wird sowohl die Übernahme von Ergebnissen der Arbeiten einer Entwicklungsphase in die jeweils nächste als auch die einheitliche Anbindung an ein (Data-) Dictionary urgiert.

Weiters ist in diesem Zusammenhang zu bedenken, daß (echte !!!) Projekte nicht nur lehrbuchmäßig linear von Phase zu Phase fortschreiten. Von einem durchgängigen Tool wird somit auch zu erwarten sein, daß die Auflösung einmal getroffener Entscheidungen in geeigneter Form unterstützt wird.

Selbstverständlich sind unter dem Durchgängigkeits-Aspekt auch Human-Interface Aspekte zu berücksichtigen (einheitliches "look & feel", gesamter Funktionsumfang unter einer einheitlichen Benutzeroberfläche).

4.2.3. Konsistenzüberprüfungen - Change Propagation

In Verbindung mit der Durchgängigkeit zwischen einzelnen Werkzeug-Komponenten sind auch Methoden zur Konsistenzerhaltung in den Ergebnissen einzelner Arbeitsschritte sowie zwischen diesen zu betrachten. Dies gilt selbstverständlich für die Entwicklung im linear fortschreitenden Sinn, in verstärkterem Maße jedoch für Änderungen, die nach einem Review an der Vorgängerversion eines Modells vorzunehmen sind.

4.2.4. Schriftliche Dokumente

Während Diagramme am Bildschirm (insbesondere wenn auf Workstations gearbeitet werden konnte) im allgemeinen recht befriedigend waren, stellte die Druckausgabe nahezu generell ein Problem dar. Dies liegt zwar zu wesentlichen Teilen an den Beschränkungen der Drucker und des A4-Ausgabe-Formats. Dennoch wird man für ordentliche Reviews - die doch zum Standard guter Software-Entwicklung gehören - auch Hardcopy-Ausgaben wünschen, die nicht durch Verwendung von Schere und Kleister entstehen. Als Mindestforderung in diese

Richtung wären wysiwyg-Previewer mit Eingreifmöglichkeiten anzusehen. Eigentlich würde man sich allerdings "intelligentere" Formen der Druckausgabe wünschen.

4.2.5. Reuse-Unterstützung

Als Ausfluß der implementierten phasengetriebenen Forward-Engineering-Methoden bieten die einzelnen Werkzeuge wenig bis keine Unterstützung für Software Reuse. Als Minimalforderungen wären hier die Unterstützung von Auswahllisten für vorhandene (alternative) (Teil-)Lösungen (projektspezifisches Reuse) anzusehen sowie die Unterstützung der Auswahl und Übernahme von Teilen von Arbeitsergebnissen in andere Projekte ("intelligentes" konsistenzerhaltendes cut & paste).

4.2.6. Team-Fähigkeit

Da reale Software-Entwicklung grundsätzlich Teamarbeit ist (und entsprechendes meist auch avisiert wird), waren wir über Defizite einiger Tools in diesem Bereich überrascht. Selbst bei den von uns eingesetzten Drei-Personen-Teams zeigte sich bereits sehr klar, daß Unterstützung von Teamarbeit grundsätzlich erforderlich ist und sich die Werkzeuge hierin deutlich unterscheiden.

5. ZUSAMMENFASSUNG

Es entspricht der Natur eines kritischen Artikels, daß mehr über Negatives gesprochen wird, denn über Positives. In der Tat war in der ersten Plenar-Zwischensitzung von jeder Gruppe in leicht abgewandelter Form zu hören: "Wir sind arm, weil wir haben das schlechteste Tool bekommen!" Jedoch änderte sich diese negative Einstellung während des Projektes deutlich, und die einzelnen Gruppen lernten sehr wohl die Vorzüge "ihres" Tools gegenüber den Eigenschaften der Vergleichstools zu schätzen.

Insgesamt blieb wohl bei allen Gruppen der Eindruck zurück, daß die in diesem Projekt geleistete Arbeit aufgrund der Anlaufschwierigkeiten wohl ohne Werkzeug mindestens ebensogut und rascher hätte durchgeführt werden können, daß sie jedoch nun, da sie mit einem komplexen Werkzeug ein umfangreiches und komplexes Projekt durchgeführt hätten, ihr nächstes Projekt vergleichbarer Größenordnung sicherlich nur wieder tool-unterstützt beginnen wollten.

Für den EDV-Verantwortlichen darf daraus der Schluß gezogen werden, daß mit der Entscheidung für ein bestimmtes Tool nicht so lange zugewartet werden sollte, bis ein Projekt ansteht, das ohne dieses Werkzeug nicht mehr bewerkstelligt werden kann, sondern

- daß man den Tool-Einstieg mit relativ kleinen und überschaubaren Projekten schaffen sollte, und
- daß man sich im Einstiegsprojekt (vielleicht sogar in den Einstiegsprojekten) noch keinerlei Effizienzsteigerung durch die Werkzeugverwendung erwarten sollte.

Allerdings hilft auch reines Trockentraining (und Zuschauen bei Profis) wenig. Die wirklich entscheidenden Fragen tauchen erst auf, wenn man alleine vor den Problemen realer Projekte steht. Wer den Mut hat, diesen Fragen früher in die Augen zu schauen, der wird später auch eher die bessere Software haben.

6. ACKNOWLEDGEMENT

Die Autoren möchten an dieser Stelle ausdrücklich auf den hohen Einsatz der vier Testteams verweisen und sich bei den Testern Karl Burghauser, Harald Buzzi, Alexander Felfernig, Ursula Fojan, Michael Fritz, Andreas Kenda, Hans-Peter Mikula, Georg Olivotto, Heinz Pozewaunig, Gerald Reicher, Michael Saringer, Manfred Wundara und Gerald Wurzer für deren Motivation und Interesse bedanken.

Weiterer Dank gebührt den Firmen ERNST&YOUNG, Wien, MID GmbH, Nürnberg, ORACLE Datenbanksysteme, Wien und AI INFORMATICS GesmbH, Linz für deren Unterstützungsbereitschaft.

7. LITERATUR

[CHEN 76] Chen P.P.: "The Entity-Relationship Model: Towards a unified view of data", ACM TODS, Vol. 1, No. 1, 1976, pp. 9-38

[DEMA 78] DeMarco T.: "Structured Analysis and System Specification", Yourdon Press, 1978

[GANE 79] Gane C., Sarson T.: "Structured Systems Analysis", Prentice-Hall, 1979

[MITT 93] Mittermeir R.T., Hochmüller E., Kienzl K., Kofler E., Steinberger H.: "Kriterien zur Auswahl eines CASE-Tools - Ergebnisse eines empirischen Vergleichs", Institut für Informatik, Univ. Klagenfurt, TR-UBWI 9/93, Sep. 1993.

[OLLE 80] Olle T.W. (ed.): "Comparative Review of Information Systems Design Methodologies: Problem Definition", IFIP WG 8.1, 1980

[OLLE 82] Olle T.W., Sol H.G., Verrijn-Stuart A.A. (eds.): "Information Systems Design Methodologies: A Comparative Review", North-Holland, 1982

[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

[SINZ 87] Sinz E.J.: "Datenmodellierung betrieblicher Probleme und ihre Unterstützung durch ein wissensbasiertes Entwicklungssystem", Habilitation, Universität Regensburg, 1987

[WARD 85] Ward P., Mellor S.: "Structured Techniques for Real-Time Systems", Yourdon Press/Prentice-Hall, 1985

[YOUR 79] Yourdon E., Constantine L.: "Structured Design", Prentice-Hall, 1979