Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 80 (1989)

Heft: 17

Artikel: Nutzen und Grenzen von Case

Autor: Sandmayr, H.

DOI: https://doi.org/10.5169/seals-903711

Nutzungsbedingungen

Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften auf E-Periodica. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Das Veröffentlichen von Bildern in Print- und Online-Publikationen sowie auf Social Media-Kanälen oder Webseiten ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. Mehr erfahren

Conditions d'utilisation

L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. La reproduction d'images dans des publications imprimées ou en ligne ainsi que sur des canaux de médias sociaux ou des sites web n'est autorisée qu'avec l'accord préalable des détenteurs des droits. En savoir plus

Terms of use

The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. Publishing images in print and online publications, as well as on social media channels or websites, is only permitted with the prior consent of the rights holders. Find out more

Download PDF: 03.07.2025

ETH-Bibliothek Zürich, E-Periodica, https://www.e-periodica.ch

Nutzen und Grenzen von Case

H. Sandmayr

Mit Case verbinden sich zurzeit (zu) hohe Erwartungen bezüglich Verbesserung der Produktivität in der Softwareentwicklung. In diesem Beitrag wird diskutiert, welche Rolle Case im Softwareentwicklungsablauf spielt, welchen Nutzen der Einsatz von Case bringen kann, welches die notwendigen Voraussetzungen sind und wo die Grenzen liegen.

On associe actuellement à Case des attentes (trop) hautes en ce qui concerne l'amélioration de la productivité dans le développement des logiciels. Dans cet article, on discute le rôle que joue Case dans le déroulement du développement de logiciels, l'utilité, les conditions à remplir et les limites de Case.

Adresse des Autors:
Dr. *Helmut Sandmayr*, dipl. Math. ETH, Infogem AG, 5401 Baden.

Mit dem Namen Case verbindet man heute hohe Erwartungen. Versprechungen lassen glauben, dass das Rezept für die Lösung der leidigen Softwareprobleme gefunden sei. Es ist verlockend, sich durch eine schnelle Investition in Case-Werkzeuge von den Sünden und Unterlassungen der Vergangenheit loskaufen zu können. Wer solche Erwartungen hegt oder den Versprechungen der Verkäufer unkritisch glaubt, läuft Gefahr, dass er eine herbe Enttäuschung erleben wird und dann in seiner Enttäuschung den wirklich möglichen Nutzen von Case vergibt.

Case hat ein echtes Potential für Verbesserungen im Entwicklungsablauf. Der mögliche Nutzen ist zwar weniger spektakulär, als ihn alle Beteiligten wünschen, er ist jedoch zu signifikant, als dass man ihn ignorieren dürfte. Die folgenden Überlegungen sollen zeigen, was man erwarten kann, wenn man richtig vorgeht.

Was ist Case?

Nachdem schon viele Autoren vergeblich versucht haben, eine allgemein akzeptierte Definition zu finden, soll hier der Versuch gar nicht erst unternommen werden. Im folgenden wird versucht, den Begriff anhand von Beispielen zu erläutern und abzugrenzen und so ein Verständnis für diesen Begriff zu schaffen.

Orientieren wir uns vorerst am Namen selbst. Die Abkürzung enthält die Komponenten *Computer Aided* und *Software-Engineering*. Der zweite Begriff, Software-Engineering, beschreibt die eigentliche Aktivität. Im IEEE-Standard 729-1983 [1] wird Software-Engineering definiert als

- das systematische Vorgehen für die Entwicklung, den Betrieb, die Wartung und die Ausserbetriebnahme von Software.

Uns interessiert hier vor allem das *systematische Vorgehen* bei den Tätigkeiten im Zusammenhang mit der Entwicklung und Wartung von Software, d.h. das geordnete Zusammenspiel von Abläufen in der Entwicklung, eingesetzten Methoden und Werkzeugen.

Den ersten Begriff, Computer Aided, und CA, kennen wir aus dem Bereich der Konstruktion und Fertigung in den Begriffen CAD und CAE. Es steht für Rechnerunterstützung von Tätigkeiten und Abläufen der Konstruktion. Analog können wir Case als Rechnerunterstützung für Software-Engineering betrachten oder, wie ein Buchtitel verspricht: «Case is Software/Automation» [2].

Die Analogie zwischen CAD/CAE und Case lässt sich weiterspinnen: gehen Sie dazu von der Annahme aus, dass Case heute dort ist, wo CAD vor etwa 10 Jahren war, und der Tatsache, dass damals die Abläufe in der Konstruktion bereits besser eingespielt waren, als es heute in der durchschnittlichen Software-Entwicklungsumgebung der Fall ist. Wir sollten aus den Enttäuschungen vieler CAD-Einführungsprojekte, insbesondere den nicht eingetroffenen Produktivitätssteigerungen, lernen und nicht die gleichen Fehler wiederholen.

Software-Engineering, der Prozess der Softwareentwicklung und -wartung soll automatisiert oder, etwas bescheidener ausgedrückt, durch Rechner unterstützt werden. Die Erfahrung zeigt, dass man nur Abläufe automatisieren kann, die man beherrscht. Und damit sind wir bei einer ersten Feststellung:

● Software-Engineering, d.h. ein geordneter Entwicklungsablauf, ist eine Voraussetzung für den wirksamen Einsatz von rechnergestützten Hilfsmitteln, sprich Case.

Oder anders ausgedrückt, Umgebungen, in denen geordnete Abläufe und ein methodisches Vorgehen in der Softwareentwicklung noch nicht etabliert sind, haben ein Problem beim Einsatz von Case (und vermutlich nicht nur dort): Was soll denn durch Rechner unterstützt werden? Hier ist eine zweite Feststellung angebracht:

● Es ist ein gefährlicher Trugschluss anzunehmen, dass durch die Einführung von Case (sprich Case-Werkzeugen) ein geordneter Entwicklungsablauf entsteht.

Zuerst muss der Entwicklungsablauf strukturiert und in Ordnung gebracht werden. Diese Systematik im Entwicklungsablauf ist notwendig, definiert sie doch die Anforderungen an die Hilfsmittel und stellt ein geordnetes Zusammenspiel zwischen verschiedenen Komponenten sicher. Um diese Anforderungen ist man spätestens dann froh, wenn man entdeckt, dass auf dem Case-Markt über 500 Case-Werkzeuge von etwa 200 Anbietern feilgeboten werden.

Tätigkeiten	Bedeutung für techn. Anwendung	Bedeutung für EDV
Strategische Planung	-	(i)
Projektleitung, -überwachung	i	i
Anforderungsspezifikation	i	i
Prototyping	(i)	i
Simulation	i	-
Entwurf	i	i
automatische Kodegenerierung	-	(+), i
Kodierung	+	+
Test	(+), i	(+), i
Debugging	+	i
Dokumentenerstellung	+	+
Konfigurationsverwaltung	(+)	i
Kenngrössen-Ermittlung	i	i

Tabelle I Wichtigste Tätigkeiten, für die Case-Werkzeuge angeboten werden

+ heute bereits eingesetzt
i interessante Verbesserungsmöglichkeiten
- nicht von Bedeutung
( ) bedingt

Tätigkeiten der Entwicklung - Case-Angebot

Sieht man sich auf diesem Markt für Case-Werkzeuge um, so findet man eine vielfältige Palette von Werkzeugen. Die Tabelle I soll einen Einblick geben, für welche Tätigkeiten Case-Werkzeuge angeboten werden. Sie ist weder vollständig noch nach bestimmten Kriterien geordnet. Da technische Projekte und EDV-Projekte verschiedene Charakteristiken, eine andere Ausprägung der einander entsprechenden Tätigkeiten haben, gibt es unterschiedliche Unterstützungsmöglichkeiten. Die einzelnen Werkzeuge sind daher für die verschiedenen Entwicklungsumgebungen von unterschiedlichem Interesse. Die Tabelle I zeigt, dass Werkzeuge zur Unterstützung der Kodierung und Erstellung von Datenbanken heute eingesetzt werden (und daher diejenigen, die echtes Software-Engineering betreiben, auch Case einsetzen. Im Mittelpunkt der Case-Diskussion stehen daher Werkzeuge für

- die Unterstützung der Spezifikation von Anforderungen sowie des Entwurfs,
- die automatische Generierung von Kode aus dem Entwurfsresultat,
- die Unterstützung der Softwareprüfung,
- die Konfigurationsverwaltung.

Klassen von Case-Tools

Man kann Case-Werkzeuge nach verschiedenen Arten klassifizieren. Für unsere Betrachtung sind zwei Kriterien interessant: das Kriterium der Methode und das Kriterium der Tätigkeiten innerhalb der Entwicklung. Bezüglich des ersten Unterscheidungsmerkmals verfügen wir über zwei Klassen von Werkzeugen:

1. *methodenunabhängige* wie z.B. Editoren, Programmgeneratoren,
2. *methodenbegleitende* wie z.B. ein Werkzeug zur Unterstützung des Entwurfs mit Mascot.

In bezug auf die unterstützten Tätigkeiten lassen sich drei Klassen von Werkzeugen unterscheiden:

1. Tätigkeits*spezifische* Werkzeuge,
2. Werkzeuge, die den *gesamten* Entwicklungsablauf abdecken (oder zumindest die wesentlichen Phasen der Entwicklung),
3. *Meta-Werkzeuge* oder besser Gerüste, in die verschiedene tätigkeitsspezifische Werkzeuge eingebunden werden können.

Die tätigkeitsspezifischen Werkzeuge unterstützen eine oder mehrere Methoden, z.B. *Structured Analysis* für die Spezifikation in einer oder mehreren möglichen Varianten oder *Mascot* für den Entwurf. Die phasenübergreifenden Werkzeuge versuchen eine durchgängige Lösung anzubieten, bei der Resultate eines Arbeitsschritts mehr oder weniger automatisch in die Resultate der nächsten Phase überführt werden können. Die Gerüste gehen davon aus, dass das Kernstück einer umfassenden Case-Lösung ein zentrales Informationssystem (Information Repository) ist, in dem jegliche für den Ablauf und das entstehende Produkt relevante Information gesammelt wird. Um diesen Kern gruppieren sich die verschiedenen Einzelwerkzeuge, die von unterschiedlichen Lieferanten kommen können. Als Lieferant kommt auch die eigene Entwicklungsumgebung in Frage, da sie spezifisches Wissen über die zu entwickelnden Produkte und die firmenspezifischen Abläufe hat und dieses Wissen in die Case-Lösung einbringen will.

Zwei Beispiele für Case

Wir wollen hier zwei Beispiele für Case kurz erörtern: Case für die Spezifikation von Anforderungen und Case für Testaufgaben. Anhand dieser Beispiele sollen der Nutzen sowie einige Grenzen gezeigt werden.

Das erste Beispiel ist wegen seiner Aktualität interessant und das zweite, weil man hier mehr Unterstützung bekommen kann, als man meint (und vielleicht auch mehr tun sollte, als man tut).

Das Interesse an Case für die Spezifikation und den Entwurf ist gross, besonders im technischen Bereich, da man hier eine Unterstützung in einem eher vernachlässigten Teil der Entwicklung erwartet. Bei vielen Entwicklern ist Case sogar ein Synonym für graphische Darstellung, den Einsatz von Workstations, die Aufstellung der Anforderungsspezifikation mittels Structured Analysis (SA) und den Entwurf mit Structured Design (SD) oder ähnlichen Methoden.

Der Einsatz von Case bei der Spezifikation von Anforderungen soll im folgenden am Beispiel des werkzeugunterstützten Einsatzes *Structured Analysis (SA)* diskutiert werden. Die Aussagen, die hier gemacht werden, gelten mit leichten Anpassungen auch für andere Methoden.

Die SA-Methode gibt Anleitungen für das Vorgehen, definiert Objekte, mit denen die Anforderungen modelliert werden, und legt die Darstellung der Objekte fest. Damit können gewisse Aspekte der Anforderungen formuliert und im Case-Tool gespeichert werden. Solche Aspekte sind typischerweise der Datenfluss und die Verarbeitung der Daten, Zustände und Ereignisse sowie Daten und Beziehungen zwischen diesen Daten.

Beim kombinierten Einsatz von SA und Case ergeben sich folgende Vorteile:

- Die Methode strukturiert das Vorgehen sowie die Darstellung der von der Methode berücksichtigten Aspekte der Anforderungen.
- Das Arbeiten mit der von der Methode vorgeschriebenen graphischen Notation wird attraktiv, da das Werkzeug die graphische Erfassung und vor allem die leichte Nachführung der gespeicherten Information bei Änderung der Anforderungen erlaubt.
- Mit Werkzeugen können die gespeicherten Daten nach verschiedenen Gesichtspunkten analysiert werden; Fragen, wie «Welche Funktionen benötigen die Daten XY?» usw., sind von einem Werkzeug leicht zu beantworten.
- Das Werkzeug kann bestimmte Konsistenz- und Vollständigkeitsbedingungen prüfen, z.B. ob alle verwendeten Datenflüsse auch im Datenlexikon definiert sind oder ob bei der Verfeinerung einer Funktion auch alle Datenflüsse bedient werden, die von dieser Funktion bedient werden müssen.
- Die gespeicherten Informationen können in Dokumentenform ausgegeben werden.

Die Erfahrungen nach einer längeren Einsatzdauer zeigen:

- Der spektakuläre Teil, die Erstellung der Graphiken, ist schnell gemacht. Mühsam wird die Ausarbeitung und Dokumentation der Details, z.B. der Einträge im Datenlexikon oder der Beschreibungen der atomaren Funktionen.
- Der Nutzen liegt nicht so sehr in der graphischen Darstellung als in den Möglichkeiten der Analyse der Einträge im Datenlexikon.

Die Grenzen der Methode(n) und Werkzeuge sind leicht ersichtlich:

- Aspekte von Anforderungen, die von der verwendeten Methode und dem Werkzeug nicht unterstützt werden, müssen wie bisher erarbeitet und dokumentiert werden. So unterstützt z.B. keine der bekannten Analysemethoden die Spezifikation von Anforderungen bezüglich Attributen wie *Benutzerfreundlichkeit, Portabilität, Effizienz* usw.; die Erfahrung zeigt, dass gerade diese Punkte einen starken Einfluss auf den Entwicklungsaufwand haben und bei der Softwareabnahme häufig zu Diskussionen Anlass geben.
- Werkzeuge können logische Fehler wie z.B. fehlende Funktionen, falsche Datendefinitionen oder unzulässige Vereinfachungen in den Anforderungen nicht finden.
- Die Konfigurationsverwaltung ist, wenn viele verschiedene Werkzeuge benutzt werden, noch umständlich.

Das nächste Beispiel soll zeigen, welche Möglichkeiten Case im Umfeld des *Testens* bietet. Dabei wird hier nur das eigentliche Testen, d.h. das Suchen nach dem *Vorhandensein* von Fehlern, und nicht das Debugging, das Lokalisieren und Beheben von Fehlern, angesprochen. Diese Unterscheidung ist wichtig. Debugging ist vor allem in der technischen Softwareentwicklung gut unterstützt. Zum Ausmerzen der Fehler bzw. zum asymptotischen Annähern an eine Lösung haben sich die Entwickler schon früh Werkzeuge gebaut. Das Testen aber, häufig die einzige Prüfmassnahme in der Entwicklung, hat bis heute eher den Charakter eines Spiels mit dem System als den einer systematischen Tätigkeit im Rahmen eines umfassenden Prüfkonzepts.

Was kann Case beim Testen bieten? Einiges. Beim Testen liegt das Produkt (das Programm) in der formalen Notation einer Programmiersprache vor. Daraus kann ein Analyseprogramm viel Information holen. So gibt es z.B. Werkzeuge, welche die Komplexität des Programms bestimmen und entsprechend der Komplexität definieren, welche Klassen von Testfällen notwendig sind, um eine bestimmte Testabdeckung des Programms sicherzustellen, z.B. eine Anweisungsabdekkung oder eine Zweigabdeckung[1]. Andere Werkzeuge bieten automatische Testausführung sowie eine Analyse der Resultate an.

Die Diskussion über den Einsatz solcher Werkzeuge führt schnell zur Erkenntnis, dass Testen etwas Komplexes ist, das geplant und methodisch durchgeführt werden muss. Vernünftigerweise wird das Testen in eine Prüfstrategie eingebunden und werden sowohl Reviews als auch Tests als Prüfmassnahmen auf verschiedenen Stufen im Entwicklungsablauf eingeplant.

Nutzen von Case

Der Nutzen von Case liegt offensichtlich nicht in einer Schnellkur für all die Probleme, die sich über einen mehr oder weniger langen Zeitraum im Bereich der Führung, des Software-Entwicklungsablaufs und der Ausbildung der Mitarbeiter angesammelt haben. Wie bei Werkzeugen aller Art können Case-Werkzeuge eine Methode, ein spezifisches Vorgehen unterstützen, indem sie gewisse Tätigkeiten erleichtern, die Qualität der Arbeit verbessern und die Resultate besser sichtbar machen.

Was heisst das konkret? Ein Textsystem macht noch nicht den Schriftsteller, eine Motorsäge noch nicht den Holzfäller aus. Beide Werkzeuge unterstützen aber den Fachmann in seiner Arbeit und helfen ihm, effizienter und besser zu arbeiten. Entsprechend macht Case aus einem ungenügend

[1] Jede einfache Anweisung wird mindestens einmal ausgeführt bzw. jeder Zweig einmal durchlaufen.

ausgebildeten Softwareentwickler keine Spitzenkraft. Werkzeuge entlasten von Routinearbeit, verhindern damit einen Teil von Fehlern und erlauben die Konzentration auf die wesentliche Arbeit.

Folgt man dieser Aussage, so liegt es auf der Hand, dass Case die Produktivität verbessern wird, und zwar die Produktivität von guten und schlechten Mitarbeitern. Aber:

● Case nivelliert den Leistungsunterschied zwischen Entwicklern nicht. Im Gegenteil, Case fördert gute Entwickler mehr und rascher als schlechte.

Wenn Case die Produktivität verbessert und Werkzeuge nicht zwischen richtigen und falschen Resultaten unterscheiden können, gilt aber auch die Aussage:

● Mit Case kann man in kürzerer Zeit grössere Fehler machen.

Erfahrungen mit dem Einsatz von Case zeigen eine Steigerung der Produktivität von 10 bis 25% für den gesamten Entwicklungsablauf. Aussagen der Hersteller, dass Faktoren von 5 oder 10 zu erwarten seien, setzen eine sehr spezielle Definition von 100% voraus!

Das richtige Vorgehen

Case-Werkzeuge evaluieren? Das ist, was die Verkäufer von Case-Werkzeugen wollen und die Manager tun, die schnell auf den Case-Zug aufspringen wollen. Damit eilt es aber nicht; der Zug ist nicht so voll, wie man glauben könnte. Es scheint, dass in den USA bisher weniger als 20% der Entwicklungsumgebungen Case in grösserem Umfang evaluieren oder gar einsetzen.

Deshalb sollte man sich die Zeit nehmen, zuerst die Frage zu beantworten, was in einer Softwareentwicklungsumgebung verbessert werden soll. Ist die Verbesserung von Produktivität, Qualität, Effizienz, Termintreue oder etwas anderes das wichtigste Ziel, und *wie* gedenkt man Verbesserungen zu messen? Sodann ist die Frage nach den eigenen Fähigkeiten angebracht. Welches sind die eigenen Stärken und Schwächen? Aus den beiden Antworten lässt sich dann ableiten, welche Massnahmen die grössten Chancen für die angestrebte Verbesserung bieten.

Das können Massnahmen sein wie z.B. die Einführung oder Verbesserung von Projektführungsmechanismen, die Ergänzung der Prüftechnik mit der Review-Technik, die Ausarbeitung von Standard-Inhaltsangaben für verschiedene Dokumentarten oder eben die Einführung von Case.

Das Case-Projekt

Hat die Analyse der Situation zu einem positiven Entscheid für Case geführt, kann das Case-Projekt in die Wege geleitet werden. Die Einführung wird nicht in wenigen Monaten über die Bühne gehen. Sie muss mit der nötigen Aufmerksamkeit im Entwicklungsalltag als Projekt abgewickelt werden. Will man den Erfolg sichern und einen möglichst grossen Nutzen aus den zu tätigenden Investitionen ziehen, so sind folgende Schritte unabdingbar:

1. Auswahl der Methoden; dazu gehören insbesondere:
   - die vorgängige Ausbildung, damit die Beteiligten, Management und Entwickler, die Auswahl durchführen können,
   - das Festlegen des Masstabs zur Bewertung des Erfolgs,
2. die Evaluation der Tools,
3. die Einführung anhand eines Pilotprojekts, vertiefte Schulung der Methoden und Training mit Werkzeugen,
4. die Bewertung der Resultate mit Anpassung der Methoden und Werkzeuge an die spezifischen Bedingungen der Entwicklungsumgebung und eventuell weitere Schulung der Methoden und Werkzeuge.

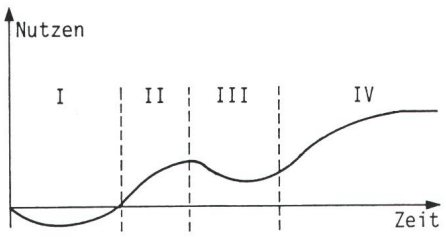
Erfahrungen über die erfolgreiche Einführung von Case lassen sich in folgender Feststellung zusammenfassen:

● Die Investitionen für die Einführung von Case betreffen vor allem die Beschaffung von Werkzeugen und die Schulung. Dabei ist der Aufwand für die Schulung gleich- bis doppelt so gross wie der Aufwand für die Werkzeuge.

Die Schulung betrifft vor allem die Methodenschulung und das Aufbauen der Erfahrungen im Umgang mit den neuen Methoden und Werkzeugen.

Eine ausführliche Diskussion des Vorgehens findet sich im Buch von Pressmann [3].

Trägt man den Nutzen von Case über der Zeitachse auf, so ergibt sich folgende Kurve:



Die Kurve zeigt nur das qualitative Verhalten; sie lässt sich in vier Abschnitte gliedern, die folgendermassen erklärt werden können:

- Im Abschnitt I ist der Nutzen negativ; die Arbeit geht langsamer vorwärts als bisher. Die Ursache liegt im Schulungsaufwand und in Fehlern beim Einsatz der Methoden und Werkzeuge im Pilotprojekt. Die Dauer dieses Abschnitts hängt stark vom Ausgangspunkt sowie vom Umfang der Neuerungen ab.
- In den Abschnitt II fällt der Abschluss des Pilotprojekts. Da man bei einem Pilotprojekt gute und motivierte Mitarbeiter einsetzt und ein Projekt auswählt, das einen günstigen Verlauf verspricht, wird der Nutzen noch innerhalb des Projekts sichtbar.
- Im Abschnitt III ist ein Rückgang des Nutzens festzustellen. Er ist durch die Einführung der Methoden und Werkzeuge in weiteren Projekten bedingt; es werden weitere Mitarbeiter ausgebildet, die nicht mehr gleich motiviert sind wie die des Pilotprojekts. Ausserdem kann man die Projekte jetzt nicht mehr nach günstigen Voraussetzungen für einen erfolgreichen Einsatz von Case aussuchen, wie das bei Pilotprojekten sinnvollerweise getan wird.
- Im Abschnitt IV nähert sich der Nutzen asymptotisch an den durch die gewählte Methoden und Werkzeuge in der spezifischen Umgebung möglichen Nutzen.

Diese Kurve ist nicht Case-spezifisch. Sie gilt in entsprechender Form auch für andere grundlegende Änderungen der Arbeitsumgebung. Wichtig ist jedoch, die einzelnen Phasen vernünftig abzuschätzen, realistische Erwartungen zu formulieren und keine unmöglichen Versprechungen zum Erlangen der Mittel für die notwendige Investition zu machen. Folgende Feststellung ist aus den obigen Ausführungen verständlich:

● Verständnis und Unterstützung des Managements sind für die Einführung von Case notwendig.

Bei der Einführung werden Probleme entstehen, Misserfolge eintreten; dann ist es unerlässlich, dass ein überzeugtes Management hinter dem Entscheid steht, bis eine echte Bewertung möglich wird.

Zwei abschliessende Bemerkungen

Case ist bei richtigem Vorgehen eine nützliche Massnahme zur Verbesserung der Softwareentwicklung. Aber man vergesse nicht:

- Software-Engineering ist das Wichtigste an Case.

Abläufe, Methoden und Werkzeuge müssen zusammenspielen. Abläufe und Methoden ändern sich langsamer als Werkzeuge, und ihre Einführung erfordert höhere Investitionen. Daher ist ihre Festlegung von grösserer Bedeutung.

Auch im Zeitalter von Case bestimmen immer noch die gleichen drei Faktoren die Produktivität:

- die Qualifikation der Mitarbeiter,
- die Effizienz der Entwicklungsschritte
- einfachere Produkte

Daher müssen Massnahmen zur Verbesserung bei diesen Faktoren ansetzen.

Case ist eine wichtige, aber untergeordnete Massnahme, die einen Beitrag zur Effizienz der Entwicklungsschritte leistet – und auch dann noch leisten wird, wenn der Name Case bereits durch ein anderes Modewort ersetzt sein wird.

Literatur

[1] Glossary of software engineering terminology. IEEE-Standard 729-1983.

[2] *C. McClure:* CASE is software automation. Englewood Cliffs/N.J., Prentice-Hall, 1989.

[3] *R.S. Pressmann:* Making software engineering happen. A guide for instituting the technology. Englewood Cliffs/N.J., Prentice-Hall, 1988.

Kennen Sie die ITG?

SEV/ASE

Die Informationstechnische Gesellschaft des SEV (ITG) ist ein *nationales Forum* zur Behandlung aktueller, anwendungsorientierter Probleme im Bereich der Elektronik und Informationstechnik. Als *Fachgesellschaft des Schweizerischen Elektrotechnischen Vereins* (SEV) steht sie allen interessierten Fachleuten und Anwendern aus dem Gebiet der Informationstechnik offen.

Auskünfte und Unterlagen erhalten Sie beim Schweizerischen Elektrotechnischen Verein, Seefeldstrasse 301, Postfach, 8034 Zürich, Telefon 01/384 91 11.

Unser Entstörungslabor

- prüft die Störempfindlichkeit und das Störvermögen,
- bestimmt Störschutz- und Schirmmassnahmen,
- kontrolliert Apparate und Anlagen auf Einhaltung der gesetzlichen Störschutzbestimmungen,
- führt Prototyp- und serienmässige Entstörungen aus,
- steht Fabrikations- und Importfirmen für fachmännische Beratung in EMV-Problemen zur Verfügung.

PRO RADIO-TELEVISION, Entstörungslabor, 3084 Wabern, Telefon 031 / 54 22 44

Une sécurité simple, efficace, économique:
Utilise la chaleur humaine pour la détection (Infrarouge)
Utilise le réseau électrique pour la transmission (Transec®)
Utilise les douilles de lampes pour l'installation.

TRANSEC®

Einfach, wirksam, preisgünstig:
Verwendet die Körperstrahlung für die Bewachung (Infrarot)
Verwendet das existierende Stromnetz für die Übermitlung (Transec®)
Verwendet die Spots- und Lampenfassungen für die Installation

electro bauer

DISTRIBUTEUR EXCLUSIF DU SPOT-ALARM POUR LA SUISSE

EXKLUSIVVERTRETUNG DES SPOT-ALARMS FÜR DIE SCHWEIZ

Allschwil	061 63 98 88
Bern	031 42 20 44
Chur	081 22 95 95
Colombier	038 41 18 18
Gd-Lancy	022 43 21 20
Lugano	091 51 39 33
Zürich	01 271 26 22

Electro Bauer AG
Elektrotechn. Artikel en gros
Lettenweg 114, 4123 Allschwil

Kompaktstationen von **ascom**, *die ausgereifte Technik mit* **Mono 2**

Die Mono 2 ist wegen ihrer niedrigen Bauhöhe zur Aufstellung in Vorgärten, Parkanlagen, Grünstreifen und an Strassenkreuzungen die perfekte Lösung. Einböschungen sind möglich. Die Schwellenhöhe an der Frontseite beträgt 0,20 m.

Die Trafostation ist «PEHLA»-geprüft und entspricht somit den höchsten Anforderungen bezüglich Personensicherheit und Materialspezifikation. Wasserdichte Rohreinführungen gehören zur Standardausrüstung.

ascom *Energieverteilung zuverlässig und kompetent*

Ascom Proelektra AG St. Gallerstrasse 71, 9500 Wil
Telefon 073 23 60 30, Telefax 073 23 17 09

ASEA BROWN BOVERI

VOTRE PARTENAIRE COMPETENT POUR:

Transformateurs de distribution

- À BAIN D'HUILE
- AU FLUIDE SILICONE
- À RÉSINE MOULÉE

IHR KOMPETENTER PARTNER FÜR:

Verteiltransformatoren

- MIT ÖLFÜLLUNG
- MIT SILIKONFÜLLUNG
- IN GIESSHARZAUSFÜHRUNG

ABB Sécheron SA
Transformateurs
CH – 1211 Genève 21
Tél. (022) 394 11 11
Tx. 22 130
Telefax (022) 34 21 94

ABB
ASEA BROWN BOVERI

CH – S 5009 88 FD