Dipl.-Ing. Karl-Heinz Haas
Allgemein beeideter und gerichtlich zertifizierter Sachverständiger

Softwarehaus im Konkurs

Bewertung von Software

I. Einleitung

Österreich hat im Jahr 2000 im Bereich IKT (Informations- und Kommunikationstechnologie) 20,3 Milliarden umgesetzt, im Vergleich zum Fremdenverkehr mit 15,5 Milliarden (das Klischee des Tourismuslandes Österreich kommt damit sehr ins Wanken) und der Land- und Forstwirtschaft mit 3,7 Milliarden .

Auf die Software entfallen dabei insgesamt 1,74 Milliarden , davon tragen österreichische Unternehmen 656 Millionen und multinationale Unternehmen 1,07 Milliarden bei.

Das Wachstum der 30 größten Softwareanbieter betrug 1999 bis 2000 durchschnittlich 29% [7], [9].

Dennoch geht jedes vierte kleine Softwarehaus nach ca. 3 Jahren in den Konkurs und jedes fünfte EDV-Unternehmen wird nicht einmal 5 Jahre alt [8].

Nur 7% der freiberuflichen Softwareentwickler sind länger als 10 Jahre selbständig [6].

Die Insolvenz eines Softwarehauses ist schlimm für die Inhaber und die Angestellten, katastrophal aber für die Kunden und Anwender. Software, die nicht gepflegt wird, kann nach kurzer Zeit unbrauchbar werden.

Dem Masseverwalter zeigt sich oft folgendes Bild: Eine „fast" fertige innovative Software eines sehr technikorientierten Softwarehauses, engagierte Entwickler und Visionen, wenig Ahnung von Betriebswirtschaft und Markt, hoher bisheriger Entwicklungsaufwand, aber kaum Kunden und Erlöse.

Stehen aber vielleicht der Markterfolg und die Gewinne kurz bevor? Ist dem Softwarehaus nur bei der Markterschließung „die Luft ausgegangen", weil die Banken die Finanzierung eingestellt haben?

An den Sachverständigen wird dann die Frage gestellt: **„Was ist die Software wert?"**

Software ist ein Wirtschaftsgut wie jedes andere. Eine Bewertung der Software als Bewertung eines beweglichen Wirtschaftsgutes sollte daher möglich sein. Für andere bewegliche Wirtschaftsgüter haben sich in Literatur und Rechtsprechung Begriffe wie Anschaffungswert, Neuwert, Zeitwert, Verkehrswert und Restwert herausgebildet und es bestehen feste Bewertungsregeln.

Solche Regeln existieren aber für die Bewertung von Software kaum. Auch Literatur ist zu diesem Thema wenig vorhanden [5], [10], [11], [12].

Die vorliegende Arbeit verwendet Verfahren und Methoden aus der betriebswirtschaftlichen Literatur und der Aufwandschätzung aus dem Bereich des Software-Engineering.

Es werden zwei zentrale Begriffe definiert:

- Weiterführungswert zu Reproduktionskosten (Wiederbeschaffungskosten der Software in gleicher Art und Güte)
- Liquidierungswert (Veräußerungserlös)

In einem abschließenden Fallbeispiel werden für ein fiktives insolventes Softwarehaus der Weiterführungswert und der Liquidierungswert in zehn Schritten hergeleitet.

II. Bewertung der Software – Grundlagen

A. Gegenstand der Bewertung

Bei der Bewertung der Software eines Softwarehauses handelt es sich um eine Schätzung von Software mit mehrfachem Nutzungs- und Verwertungsrecht.

Geschäftszweck eines Softwarehauses ist die Entwicklung, die Pflege und der Vertrieb einer meist branchenorientierten Anwendungssoftware. Zusätzlich werden noch Dienstleistungen rund um das Softwareprodukt, wie Installation, Unterstützung bei der Inbetriebnahme, Schulung und Einweisung angeboten.

B. Zweck der Bewertung

Zweck der Bewertung ist die Ermittlung des Wertes der Software bei

- Weiterführung oder
- Liquidierung des Softwarehauses.

C. Bereiche der Bewertung

Die Bewertung der Software muss – je nach Art und Umfang der Software, sowie Zweck der Bewertung – folgende Bereiche besonders berücksichtigen:

- Technik
- Markt
- Betriebswirtschaftliches Umfeld
- Rechtliches Umfeld

1. Technik

- Entwicklungsaufwand
- Leistungsfähigkeit der Software
- Qualität der Software
- Qualität des Software-Entwicklungs-Prozesses (Software Engineering nach Stand der Technik)
- Pflegbarkeit
- Einhaltung von Standards und Normen
- Rechtliche Anpassung
- Dienstleistungen

2. Markt

- Marktpreise
- Potentielle Kunden
- Mitbewerber
- Schätzung der Marktentwicklung

3. Betriebswirtschaftliches Umfeld

- Bisherige Erlöse aus Lizenzen, Pflegeverträge und Dienstleistungen

- Wert der abgeschlossenen Pflegeverträge
- Aufbau- und Ablauforganisation des Softwarehauses
- Standort (Nähe zu Kunden und zu Entwicklern)
- Personal
- Finanzierung

4. Rechtliches Umfeld

Software genießt nach österreichischem Recht urheberrechtlichen Schutz. Insbesondere gelten für Computerprogramme die Sondervorschriften des Abschnittes Via. UrhG (§ 40 a–f UrhG) und für Datenbankwerke die Sondervorschriften des Abschnittes VIb. UrhG (§ 40 f–h UrhG).

Urheberrechtlich relevant können also sein:

- Verträge mit Kunden, die diese Software erworben haben
- Verträge mit Personen, die diese Software (mit-)entwickelt haben

Folgende Problemstellungen sind möglich:

- Hinterlegungsvereinbarungen für den Quellcode (bekannt auch als *„Escrow Agreement"*), gekennzeichnet durch zwei Bedingungen (Herausgabebedingungen – meist Handlungsunfähigkeit, Rechte)
- Verträge mit sogenannten „freien" Mitarbeitern (Fiktion des § 40 b UrhG gilt nur für Dienstnehmer!)
- Aufträge an andere Softwarehäuser (analog „freie" Mitarbeiter)
- Sonstige Verträge über Übertragung von Urheberrechten

D. Definition von Software

Unter **Software** (vgl. auch § 40a Abs 2 UrhG) versteht man alle immateriellen Komponenten eines EDV-Systems, also

- Programme und
- Dokumentation

Programme sind

- Quellcode („was der Mensch versteht" – im Quellcode ist das Know-how verankert) und
- Maschinencode oder ausführbares Programm („was die Maschine versteht")

Zur **Dokumentation** gehören

- Entwicklungsdokumentation und
  - o Lastenheft mit den primären Anforderungen
  - o Pflichtenheft (Systemspezifikation), bestehend aus Softwarespezifikation und Imp-lementierungskonzept – definiert das System (*„WAS"*), auch als FACHLICHE Spezifikation bezeichnet
  - o Lösungsspezifikation (mit Entwurfsspezifikation) – entwirft das System (*„WIE"* und *„WOMIT"*), auch als TECHNISCHE Spezifikation bezeichnet
  - o Unterlagen über Qualitätssicherung und Testspezifikation (Testfälle)
- Programmdokumentation und
  - o Kommentierung des Quellcodes (wie hat der Entwickler die technische Spezifikation auf Papier in den Quellcode und schlussendlich in Anweisungen an den Computer übertragen?)
- Benutzerdokumentation und
  - o Installationsanleitung
  - o Systemhandbuch
  - o Benutzerhandbuch
- Onlinehilfe im Programm

Der Erwerber von **Standardsoftware-Produkten** erhält in der Regel:

- Ausführbares Programm mit Onlinehilfe auf Datenträger
- Benutzerdokumentation auf Papier
- Dienstleistungen

E. Verfahren und Methoden der Bewertung

Unabhängig vom Zweck der Bewertung muss die Software in obiger Definition als Ganzes bewertet werden.

Für eine Weiterentwicklung und Verwertung der Software sind insbesondere der Quellcode mit den entsprechenden Urheberrechten und die Dokumentation notwendig!

In der betriebswirtschaftlichen Literatur und Praxis unterscheidet man bei Unternehmensbewertungen folgende Verfahren ([1] Seite 232 ff):

- Gesamtbewertung
  - o Ertragswertverfahren
  - o Discounted Cashflow-Verfahren
- Einzelbewertung – Substanzwert zu
  - o Reproduktionswerten
  - o Liquidationswerten
- Mischverfahren
  - o Mittelwertverfahren
  - o Übergewinnverfahren

Im Fall der Liquidation, ist das Substanzwertverfahren relevant und nach [1] Seite 250 auch das einzig richtige. Das Substanzwertverfahren kommt auch noch bei bestimmten Fällen von Verlustunternehmen zur Anwendung.

Die Ermittlung des Wertes der **Software** (nicht der Gesamtfirma!) wird bei

Weiterführung des Softwarehauses
durch das Substanzwertverfahren durchgeführt.
Es wird ein Weiterführungswert zu Reproduktionskosten (Wiederbeschaffungskosten der Software in gleicher Art und Güte) ermittelt

Liquidierung des Softwarehauses
durch das Substanzwertverfahren durchgeführt.
Es wird ein Liquidierungswert (Veräußerungserlös) ermittelt

Im Falle der Weiterführung ist es empfehlenswert einen **Businessplan** für das Softwarehaus zu erstellen, mit folgenden Inhalten ([1] Seite 213 ff):

- Executive Summary
- Unternehmensgegenstand / Idee
- Marktanalyse
- Marketingkonzept
- Unternehmensprofil
- SWOT-Analyse und zukünftige Unternehmensentwicklung
- Risikoanalyse
- Finanzplanung und Cashflow-Prognose
- Zeit- und Aktivitätsplan
- Anhang

Auf Basis des Businessplanes sollte dann eine **Discounted Cashflow-Analyse** (DCF) durchgeführt werden.

Businessplan und DCF-Analyse sind Grundvoraussetzungen einer **Finanzierung**, egal ob mit Eigenkapital oder mit Fremdkapital.

Neben dem „objektiv" errechneten Wert der Software sind noch externe Markteinflüsse wie Angebot und Nachfrage sowie das Timing der Transaktion und persönliche Gründe bei den Verhandlungspartnern von Relevanz.

F. Grundlagen der Aufwandschätzung von Software

Die Vorgehensweise der Aufwandschätzung von Software kann wie folgt beschrieben werden ([2] Seite 97 ff):

- Die Schätzmethoden liefern als Ergebnis eine **Maßzahl für die Größe** (Umfang) durch Zählung
- Auf Basis einer Metrik wird daraus der **Aufwand** ermittelt
- Stehen mehrere Schätzmethoden zur Verfügung, kann ein **Mittelwertverfahren** zur Berechnung des Aufwandes angewendet werden
- Mit der Prozentsatzmethode kann der Gesamtaufwand auf die einzelnen **Phasen** eines Projektes verteilt werden
- Mittels Verrechnungssätzen je Phase werden **Kostengrößen je Phase** und eine **Gesamtkostengröße** gewonnen

Es existieren zwei grundsätzliche Ansätze für die Aufwandschätzung von Software:

- Umfang der funktionalen Vorgaben aus Sicht des Anwenders, die den genauen Inhalt der Software beschreiben – aus funktionaler Sicht gemessen
- Programmumfang (LOC - Lines of Code) - aus technischer Sicht gemessen

De-facto-Standard bei der Messung des funktionalen Umfanges ist die **Function-Point-Methode** [3].

Für die Bewertung der Software nach dem Programmumfang wird eine LOC-Methode (Lines of Code) verwendet. Bekannt ist die weitverbreitete **COCOMO-Methode** (Constructive Cost Model) oder bei ALT-Systemen ohne Dokumentation das sogenannte **Backfiring**.

Liegen Ergebnisse von mehreren Methoden vor, kann der Aufwand durch ein Mittelwertverfahren approximiert werden.

Mit der **Prozentsatzmethode** wird eine relative Aufteilung des Aufwandes bzw. der Kosten auf einzelne Phasen vorgenommen.

Durch Festlegung von **Verrechnungssätzen** wird aus dem Aufwand eine Kostengröße gewonnen.

Der Aufwand für ein Software-Produkt ist aber nicht nur eine Funktion der Größe, sondern eine Funktion vieler Einflussfaktoren, wie Erfahrung der Mitarbeiter, Klarheit der Anforderungen, Prozessreife des Softwarehauses, ...).

Die Schätzproblematik zusammengefasst ([2] Seite 17):

- Schätzen hat mit Ungewissheit zu tun
- Beim Schätzen werden Informationen zum Schätzobjekt benötigt
- Schätzen wird oft mit Verhandeln verwechselt!

G. Methoden der Aufwandschätzung von Software

1. Function-Point-Methode

Die Function-Point-Methode wurde 1979 von der Firma IBM entwickelt.

Auf Grundlage von fünf definierten Basis-Funktions-Komponenten werden nach einem bestimmten Verfahren mit unterschiedlichen Gewichtungsfaktoren Function Points gezählt. Diese Function Points werden anschließend mit Hilfe einer statistischen Methode in Aufwand umgerechnet. Dabei wird eine Korrelation zwischen Function Points und Aufwand unterstellt. Anhand einer Korrelationskurve (Aufwand- oder Produktivitätskurve), die aus der firmeneigenen Erfahrungsdatenbank abgeschlossener Projekte resultiert, wird der Aufwand ermittelt.

Meist fehlt aber eine firmeneigene Korrelationskurve. In diesem Fall kann ersatzweise auf Kurven, die in der Literatur publiziert wurden, zurück gegriffen werden, zB auf die Kurve der IBM ([2] Seite 101), die in einer Formel ausgedrückt, lautet:

$$FP = 25{,}649 \times \left(PM^{0{,}7752}\right)$$ **Gleichung 1**

FP .... Anzahl der Function Points
PM ... Anzahl der Personen-Monate

Bei bekannten Function Points (FP) kann nun der Aufwand in Personen-Monaten (PM) berechnet werden:

$$PM = \sqrt[0{,}7752]{\frac{FP}{25{,}649}}$$ **Gleichung 2**

Die Function-Point-Methode verlangt aber nach dem Entwurf ISO/IEC 14143, dass eine funktionale Anforderung aus Sicht des Anwenders vorliegt. Es muss also die FACHLICHE Spezifikation vorliegen.

In der Praxis wird aber gerade bei Softwarehäusern im Konkurs keine fachliche Spezifikation vorliegen. Eine Ermittlung der Function Points wird daher nicht möglich sein.

2. Grundsätze der LOC-Methoden

Die LOC (Lines of Code) Methoden gehen davon aus, dass eine sehr hohe Korrelation zwischen der Größe einer Software (ausgedrückt in Lines of Code) und dem Aufwand (ausgedrückt in Personen-Monaten) herrscht.

Allerdings sind die unterschiedlichen Zählweisen bei der LOC-Metrik zu berücksichtigen ([4] Seite 34 f).

In der Praxis ist es am einfachsten, die physikalischen Zeilen, wie sie sich bei Verwendung eines Editors ergeben, zu zählen. Üblicherweise verwendet man nur die Begrenzungen der Konstrukte der eingesetzten Programmiersprache (delimiter) zur Zählung. In der Literatur werden Unterschiede in der Größenordnung von 2,5 bis 5,0 genannt ([4] Seite 34 setzt 247% an).

$$LOC = \frac{Anzahl\ Physikalische\ Zeilen}{2{,}47}$$ **Gleichung 3**

Da die Programmiersprachen unterschiedlich mächtig sind, müssen die LOC noch normalisiert werden. Man verwendet dazu Assembler mit der Mächtigkeit 1 *(Assembler Equivalent, AE).*

Einige populäre Programmiersprachen weisen folgenden *language level* auf ([4] Seite 40):

C	2,5
COBOL	3,0
PASCAL	3,5
BASIC	5,0
Query Languages	20,0
Spreadsheet Languages	50,0

Somit berechnet sich – abhängig von der verwendeten Programmiersprache und ihrem language level – das Assembler Equivalent von LOC:

$$LOC_{AE} = \frac{LOC}{AE}$$ **Gleichung 4**

3. COCOMO-Methode

Eine bekannte LOC-Methode ist die COCOMO-Methode (Constructive Cost Model). Sie berechnet direkt aus den LOC (gemeint sind im folgenden LOC_{AE}) den Aufwand in Personen-Monaten (PM), allerdings in Abhängigkeit von der Größe des Projektes und der Komplexität, nach folgender allgemeiner Formel ([2] Seite 115):

$$PM = a \times KLOC^{b}$$ **Gleichung 5**

wobei **a** und **b** aus folgender Tabelle entnommen werden können ([2] Seite 115):

Komplexität / Größe	< 50 KLOC	50 – 300 KLOC	> 300 KLOC
einfach	a=2,4 b=1,05	a=3,2 b=1,05	a=3,2 b=1,05
mitteischwer	a=3,0 b=1,12	a=3,0 b=1,12	a=3,0 b=1,12
komplex	a=3,6 b=1,20	a=2,80 b=1,20	a=2,8 b=1,20

4. Backfiring

Das Backfiring ist umstritten, weil davon ausgegangen wird, dass ein technisches Maß (Lines of Code) auch ein funktionales Maß (Function Points) ist und umgekehrt. Backfiring ist aber oft die einzige Möglichkeit, die Funktionalität von ALT-Systemen zu erfassen oder von Software ohne Dokumentation ([2] Seite 104 ff).

Ausgehend von der **Codegröße in LOC** (gemeint sind im folgenden LOC_{AE}) und **Codekomplexität** werden Rückschlüsse auf die Funktionalität der Software gezogen.

Die Frage, wie viel Quellcodezeilen nun in einer bestimmten Programmiersprache benötigt werden, um einen Function Point zu kodieren *(durchschnittliche Expansionsrate)*, beantwortet folgende Tabelle für einige populäre Programmiersprachen:

C 128
C++ 55
COBOL 107
BASIC 35
Query Languages 13
Spreadsheet Languages 6

Somit lassen sich die benötigten Function Points (FP) einer Software mit einer bestimmten Anzahl von Codezeilen (LOC) und einer bestimmten Komplexität (Anpassungsfaktor oder engl. adjustment multiplier) annähern:

$$FP = \frac{LOC}{durchschnittliche\ Expansionsrate \times Anpassungsfaktor}$$

Gleichung 6

wobei der Anpassungsfaktor sich aus Problem-, Code- und Datenkomplexität berechnet und zwischen 0,70 und 1,30 liegt.

Besitzt das Softwarehaus eine Korrelationskurve, kann aus dieser der Aufwand in Personen-Monaten (PM) abgelesen werden. Ersatzweise kann die Korrelationskurve von IBM verwendet werden und PM mit **Gleichung 2** berechnet werden.

5. Mittelwert-Methode

Umstritten ([1] Seite 251), weil ohne jegliche logische oder theoretische Begründung – dennoch in der Praxis überraschend tauglich, kann der Aufwand nach der Anwendung verschiedener Methoden durch ein Mittelwertverfahren angenähert werden.

Setzt man in der Praxis die Methoden COCOMO und Backfiring ein, ergibt das arithmetische Mittel folgenden Aufwand in Personen-Monaten (PM):

$$PM = \frac{PM_{COCOMO} + PM_{Backfiring}}{2}$$ **Gleichung 7**

6. Prozentsatz-Methode

Es ist Stand der Technik, dass die Softwareentwicklung nach einem Phasenmodell abläuft. Der Gesamtaufwand wird abhängig von der Projektgröße im Durchschnitt wie folgt auf die einzelnen Phasen aufgeteilt ([4] Seite 54):

Phase	Aufwand in %
Analyse der Anforderungen	6 %
Grobentwurf	15 %
Feinentwurf	23 %
Implementierung (Codierung und Modultest)	35 %
Integrationstest	21 %

Bemerkenswert sind folgende Tatsachen:
- Der Anteil der reinen Programmierung (ohne Test und Quellcode-Kommentierung) liegt bei ca. 25 % des Aufwandes
- Der Anteil der gesamten Dokumentation liegt bei ca. 50% des Aufwandes

7. Verrechnungssätze

In der Regel wird als Einheit für den Aufwand der Mannmonat (MM) verwendet, die geschlechtsneutrale Bezeichnung Personenmonat (PM) setzt sich aber immer mehr durch. Davon abgeleitet werden auch andere Einheiten für den Aufwand verwendet und wie folgt umgerechnet ([4] Seite 51). Die Zahlen berücksichtigen bereits Urlaub und Krankheitszeiten, sind aber länderspezifisch, für Österreich gilt:

1 Personentag (PT) = 8 Stunden
1 Personenwoche (PW) = 5 PT = 40 Stunden
1 Personenmonat (PM) = 4 PW = 20 PT = 160 Stunden
1 Personenjahr (PJ) = 10 PM = 40 PW = 200 PT = 1600 Stunden

Während der Phasen der Softwareentwicklung sind unterschiedliche Qualifikationen notwendig und daher müssen auch phasenspezifische Verrechnungssätze angewendet werden.

Somit berechnen sich die Kosten einer Phase i:

$$Kosten_i = Aufwand_i \times Stundensatz_i$$ **Gleichung 8**

und die Gesamtkosten, wenn n die Anzahl der Phasen ist:

$$Gesamtkosten = \sum_{i=1}^{n} Kosten_i$$ **Gleichung 9**

Ein Anhaltspunkt für die Stundensätze (exklusive der gesetzlichen Mehrwertsteuer), die am österreichischen Markt je nach Tätigkeiten verlangt und bezahlt werden, kann folgende Tabelle sein (auf Basis der Daten von [6]):

Phase	Tätigkeit	Stundensatz in €
Analyse der Anforderungen	Consulting	90 €
Grobentwurf	Consulting	90 €
Feinentwurf	Softwaredesign	60 €
Implementierung (Codierung und Modultest)	Programmierung	50 €
Integrationstest	Qualitätssicherung	40 €

III. Fallbeispiel

Ein kleines Softwarehaus stellt seit Jahren eine sehr marktenge branchenspezifische Anwendungssoftware her, hat kaum Erfolg und wird schließlich insolvent.

Nach dem **Capability Maturity Model,** einem Standard-Modell zur Einschätzung der Prozess-Reife, befindet sich das Softwarehaus wie die Mehrheit aller Softwarehäuser auf der ersten (**Initial**) von fünf möglichen Stufen, die so definiert ist ([4] Seite 129):

> *Ein Prozess zur Erstellung der Software ist nicht definiert. Erfolge sind zwar möglich, beruhen jedoch auf den Anstrengungen einzelner Mitarbeiter und sind eher zufällig.*

Die Dokumentation ist völlig unbrauchbar, der Quellcode nach dem Motto: *„Ein guter Quellcode dokumentiert sich selbst"* völlig unkommentiert. Mit einem Wort: Das Softwarehaus betreibt Programmierung als Künstlertum und nicht als Ingenieursdisziplin.

Was ist nun der innere, der technische Wert der bezüglich Komplexität *einfachen* Software? Was kann bei Veräußerung für die Software am Markt erlöst werden?

A. Wiederbeschaffungswert

Es wird der Wiederbeschaffungswert zu Reproduktionskosten der Software nach gleicher Art und Güte nach den vorgestellten Verfahren und Methoden in zehn Schritten ermittelt:

1. Ausgangspunkt und leicht zu ermitteln: der Quellcode umfasst ca. 400.000 physikalisch in einem Editor gemessene Codezeilen.
2. Die verwendete Programmiersprache ist BASIC.
3. Die Umrechnung in Sprachkonstrukte ergibt ca. 162.000 LOC (Gleichung 3)
4. Berücksichtigt man den language level der Programmiersprache BASIC (dieser beträgt 5,0) ergibt sich das Assembler Equivalent von LOC mit ca. 32.400 (Gleichung 4)
5. Die COCOMO Methode ergibt bei einfacher Software und < 50 KLOC mit dem Faktor a=2,4 und b=1,05 und der Gleichung 5 ca. einen Aufwand von 93 Personenmonate.
6. BACKFIRING ergibt bei der Programmiersprache BASIC (durchschnittliche Expansionsrate = 35) und einem angenommenen Anpassungsfaktor von 1 nach der Gleichung 6 ca. 926 Function Points.
7. Aus der Korrelationskurve von IBM und der Gleichung 2 ergibt sich ca. ein Aufwand von 102 Personenmonate.
8. Das arithmetische Mittel beider Verfahren ergibt schlussendlich einen Aufwand von 97,5 Personenmonate oder davon abgeleitet 15.600 Stunden.
9. Auf die einzelnen Phasen entfallen somit folgende Aufwände und Kosten:

   Analyse der Anforderungen .....936 Stunden .....84.240 €
   Grobentwurf .....2.340 Stunden .....210.600 €
   Feinentwurf .....3.588 Stunden .....215.280 €
   Implementierung .....5.460 Stunden .....273.000 €
   Integrationstest .....3.276 Stunden .....131.040 €

   Die Gesamtkosten betragen somit 914.160 € exklusive der gesetzlichen Mehrwertsteuer.
10. Ist die Entwicklerdokumentation nun völlig unbrauchbar, sind die Dokumente der Phasen Analyse der Anforderungen, Grobentwurf und Feinentwurf nicht vorhanden, so reduziert sich der Wert der Software erheblich auf 404.040 €.

B. Rechtsmängel

In der Praxis häufig auftauchende Probleme sind Aufträge an andere Softwarehäuser oder die Beschäftigung sogenannter freier Mitarbeiter – wenn, was meist zutrifft, die Urheberrechtsfrage nicht vertraglich vereinbart ist.

In beiden Fällen gilt die Fiktion des § 40 b UrhG nicht! Diese regelt nur im Falle eines Dienstnehmers, der in Erfüllung seiner dienstlichen Obliegenheit (zB angestellter Programmierer) ein Computerprogramm schafft, dass dem Dienstgeber hieran ein unbeschränktes Werknutzungsrecht zusteht.

Man muss also davon ausgehen, dass an bestimmten Teilen der Software, andere Personen zumindest Miturheberschaft anmelden.

Zwei Lösungsansätze bieten sich an:
- Nachträgliche vertragliche Vereinbarung und Abgeltung, falls möglich
- Entfernung des urheberrechtlich bedenklichen Codes und Neuprogrammierung

Der Aufwand bzw. die Kosten dieser Software-Teile können ermittelt werden durch:
- Werklohn und Aufwandsführung
- Aufwandschätzung

C. Liquidierungswert

Bei der Schätzung des Veräußerungserlöses müssen ausgehend von den Wiederbeschaffungskosten neben der Technik insbesondere auch das Marktumfeld und die Personalsituation gesehen werden.

Software, die nicht gepflegt wird, kann nach kurzer Zeit praktisch wertlos sein. Beispiele aus der nahen Vergangenheit wie Jahr-2000-Problem und die Umstellung auf den Euro belegen dies eindrucksvoll.

1. Marktumfeld

Welche Gründe kann es nun geben, dass Software übernommen wird:
- Übernahme durch Mitbewerb
  - o Aufbau eines neuen Geschäftsfeldes
  - o Integration des Know-how in eigenes Produkt

- o Übernahme der Entwickler
- o Marktbereinigung und Übernahme der Kunden

- Übernahme durch (Groß-)Anwender oder Vereinigungen von Anwendern
  - o Beispiel: Anwender haben sehr viel Zeit und Geld investiert und ihr Unternehmen an die Software angepasst (eventuell auch durch Individualerweiterungen). Die Anwender sind grundsätzlich mit der Software zufrieden, eine Ablöse der Software ist unerwünscht und kurzfristig auch unmöglich

2. Personalsituation

Die Personalsituation bei Softwarehäusern im Konkurs ist meist wie folgt:

- Das fachliche Know-how liegt bei wenigen Personen (und dort nur im Kopf), meist sogar bei den Geschäftsführer-Gesellschafter.
- Das technische Know-how liegt meist bei wenigen Personen (und dort nur im Kopf), den Programmierern des Kernteams.
- Bei Liquidierung ist davon auszugehen, dass sowohl das fachliche als auch das technische Know-how verloren gehen. Was bleibt ist das ausführbare Programm und der unkommentierte Quellcode.
- Andererseits haben die bisherigen Entwickler meist Interesse daran, weiterzuarbeiten, wenn die Finanzierung und somit auch die Gehälter in der Zukunft gesichert sind.

3. Schätzung in der Praxis

Die Höhe des Liquidierungswertes hängt vom Einzelfall ab und vom Verhandlungsgeschick des Veräußerers.

In einem konkreten Fall wurde ausgehend von den Wiederbeschaffungskosten und der Bedeutung von Technik, Markt und Personal im Verhältnis:

Technik . . . . . .50 %
Markt . . . . . .25 %
Personal . . . . . .25 %

und unter Herausrechnung der Mängel der Dokumentierung und der Behebung der Rechtsmängel der Liquidierungswert nach obiger Aufteilung geschätzt.

Im Fallbeispiel ergäbe dies einen Liquidierungswert von 202.020 € exklusive der gesetzlichen Mehrwertsteuer.

Literatur

[1] *Leitinger/Strohbach/Schöfer/Hummel*, Ventura Capital und Börsengänge, Manz 2000
[2] *Bundschuh/Fabry*, Aufwandschätzung von IT-Projekten, mitp Verlag 2000
[3] *Hurten Robert*, Function-Point Analysis, Expert-Verlag 1999
[4] *Thaller Georg Erwin*, Software-Metriken, Heise Verlag 1994
[5] *Otto/ZimmermannA/VißnerA/Vanders/Bühler*, Bewertung von EDV- und Elektronik-Systemen, Verlag Dr. Berndt Wißner 1995
[6] *CW-Online, MyFreelancer*, Studie im Zeitraum 17. 10.–19. 11. 2001 in Österreich, www.myfreelancer.at
[7] *Krumpak Günther*, IT-Business in Österreich 2002, Österreichische Computergesellschaft 2002
[8] *Jaburek Walter*, Handbuch der EDV-Verträge, Verlag Medien & Recht, 3. Auflage, 2000
[9] Homepage der Wirtschaftskammer Österreich www.wko.at
[10] *Juri Herbert*, Software im Konkurs, EDV & Recht 11/1991, Seite 120 ff
[11] *Praetorius/Siegel*, Softwarebewertung, Computer & Recht 1991, Seite 496 ff
[12] *Paulus Christoph*, Der Konkurs des Softwarehauses, Computer & Recht 1987

Korrespondenz:
Dipl.-Ing. Karl-Heinz Haas
Informatikbüro Lienz
Allgemein beeideter und gerichtlich zertifizierter Sachverständiger
9900 Lienz, Josef E. Plonerstraße 11
Telefon 04852-61971
Telefax 04852-61971-4
E-Mail haas@ibl.at
Homepage: www.ibl.at

Ein gesegnetes Weihnachtsfest und ein erfolgreiches neues Jahr
wünscht allen Mitgliedern und ihren Familienangehörigen
die Präsidien des Hauptverbandes der
allgemein beeideten und gerichtlich zertifizierten Sachverständigen
und der Landesverbände

DIE REDAKTION UND ANZEIGENVERWALTUNG
DER FACHZEITSCHRIFT „DER SACHVERSTÄNDIGE"
SCHLIESSEN SICH DIESEN WÜNSCHEN AN