NETZEFFEKTE, WECHSELKOSTEN UND CUSTOMER LOCK-IN AUF SOFTWAREMÄRKTEN UND DAS URHEBERRECHT

Ralf Blaha

Rechtsanwalt, Heid Schiefer Rechtsanwälte,
Niederlassung Domplatz 1, 9020 Klagenfurt am Wörthersee, AT
Kanzleisitz Landstraßer Hauptstraße 88/2–4, 1030 Wien, AT
blaha@heid-schiefer.at, www.heid-schiefer.at

Schlagworte: ***Netzeffekte, Skaleneffekte, Kompatibilität, Wechselkosten, Software-Urheberrecht, Dekompilierung, IT-Vertragsgestaltung, Customer Lock-In***

Abstract: *Softwaremärkte sind von ökonomischen Besonderheiten, wie z.B. Netzeffekten und Pfadabhängigkeiten (hohe Kosten des Abgehens von einer getroffenen Systementscheidung), geprägt. Die gesetzlichen Instrumente des Software-Urheberrechts reichen nicht aus, Lock-In-Situationen zu verhindern, weshalb im Rahmen der IT-Vertragsgestaltung Vorsorge zu treffen ist.*

1. Einleitung

Die Problematik der Pfadabhängigkeit (Customer Lock-In) für Nutzer von IT-Systemen soll anhand von zwei Beispielen aufgerissen werden:

1.1. Beispiel #1: Lock-In trotz vereinbarter Sourcecodehinterlegung

Lock-In-Situationen werden nur selten gerichtsanhängig, da die gerichtliche Geltendmachung von Ansprüchen aus IT-Projekten infolge ihrer Komplexität i.d.R. mit hohen Kosten und einem erheblichen Prozessrisiko verbunden ist. Über den Umweg des Vergaberechts kommt aber doch dann und wann eine solcher Sachverhalt an die Öffentlichkeit:

Das Vergaberecht kennt eine Regelung, welche Lock-In-Situationen treffend beschreibt. § 30 Abs. 2 Z. 2 BVergG[1] erlaubt die Vergabe eines Auftrages in einem Verhandlungsverfahren mit nur einem Bieter (also ausschreibungsfrei), «*wenn der Dienstleistungsauftrag aus technischen oder künstlerischen Gründen oder auf Grund des Schutzes von Ausschließlichkeitsrechten nur von einem bestimmten Unternehmer ausgeführt werden kann*».

Ein öffentlicher Auftraggeber hatte vor, unter Berufung auf diese Regelung einen Vertrag über die Wartung und Weiterentwicklung eines e-Shops ausschreibungsfrei zu vergeben, da der alte und neue Dienstleister das e-Shop-System entwickelt hatte und Inhaber und Alleinverfügungsberechtigter über den Sourcecode war. Ein Mitbewerber focht diese beabsichtigte Vergabe an, die damit Gegenstand eines Nachprüfungsverfahrens wurde.

Im Nachprüfungsverfahren kam hervor, dass der Altvertrag für den Auftraggeber umfangreiche urheberrechtliche Verwertungsrechte im Hinblick auf die für den Auftraggeber vorgenommenen projektspezifischen Erweiterungen und Individualentwicklungen vorgesehen hatte. Der Vertrag sah jedoch auch vor, dass der Auf-

[1] Bundesvergabegesetz 2006, BGBl. I 17/2006 i.d.F. BGBl. I 128/2013.

tragnehmer diese Individualentwicklungen in den Standard der Software übernehmen konnte, was laut dem Vertrag den Rechteerwerb des Kunden aufhob. Der Auftragnehmer erklärte im Nachprüfungsverfahren übereinstimmend mit dem Auftraggeber, alle Adaptierungen in den Standard übernommen zu haben und dass dem Auftraggeber daher keine Verwertungsrechte (wie z.B. das Recht zur Bearbeitung) zustünden. Weiter stellte sich heraus, dass der Altvertrag zwar eine Hinterlegung des Sourcecodes der Software vorsah, der Auftraggeber jedoch aus Kostengründen auf eine Hinterlegung verzichtet hatte und daher faktisch keinen Zugriff auf den Sourcecode hatte. Weiter war die Wartung der Software auch vertraglich nicht als Herausgabefall vorgesehen.

Ergebnis des Nachprüfungsverfahrens war daher, dass die Leistungen nur vom alten und neuen Vertragspartner, der alleiniger Inhaber des Sourcecodes war, erbracht werden konnten.[2] Der Auftraggeber hatte gesiegt und konnte den Auftrag ausschreibungsfrei vergeben. Wirtschaftlich gesehen hatte er jedoch durch seine Vertragsgestaltung und Verhalten danach eine lupenreine Lock-In-Situation herbeigeführt: Er war technisch und immaterialgüterrechtlich von seinem Vertragspartner abhängig, der Wettbewerb war ausgeschaltet und der Vertragspartner konnte (vermutlich) die wirtschaftlichen Bedingungen diktieren.

1.2. Beispiel #2: Lock-In trotz Verfügung über den Sourcecode

Beim zweiten Beispiel hatte sich der Auftraggeber tatsächlich gegen eine Lock-In-Situation abzusichern versucht: Hier sah der Vertrag ein umfassendes urheberrechtliches Werknutzungsrecht, welches insbesondere die Weiterentwicklung durch Dritte einschloss, und die Übergabe des Sourcecodes, die auch tatsächlich erfolgte, vor.

Als sich der Auftraggeber jedoch entschied, den ursprünglichen Wartungsvertrag mit dem Auftragnehmer zu kündigen und einen Dritten mit der Wartung und Weiterentwicklung zu beauftragen, stellte er fest, dass dies nur zu sehr hohen Kosten möglich ist. In der Folge entspann sich eine Auseinandersetzung, in der der Auftraggeber mit mangelnder Qualität und Wartbarkeit des Sourcecodes argumentierte und erklärte, dass die Software völlig neu entwickelt werden müsste und der Auftragnehmer entgegen hielt, dass vertraglich diesbezüglich keine Anforderungen vereinbart worden waren und ein besonderes Ausmaß an Wartungsfreundlichkeit keine gewöhnlich vorausgesetzte Eigenschaft von Software sei.

Der Auftraggeber hatte somit zwar alle für die Wartung und Weiterentwicklung erforderlichen urheberrechtlichen Nutzungs- und Verwertungsrechte, diese nutzten ihm jedoch nichts, da die Wartung und Weiterentwicklung durch einen Dritten langfristig teurer als eine Neuentwicklung war.

2. Ökonomische Besonderheiten von Software

Nunmehr wird untersucht, welche Besonderheiten Software aufweist, die derartige Situationen verursachen können:

2.1. Netzeffekte

Bei Software dreht sich das Gesetz von Angebot und Nachfrage ins Gegenteil: Der Wert von Software steigt nicht mit ihrer Knappheit, sondern sie wird umso wertvoller, je weiter sie verbreitet ist.[3] Denn mit der zunehmenden Verbreitung

[2] BVA 3. Januar 2011, N/0075-BVA/14/2010-56.

[3] Economides, Competition Policy in Network Industries: An Introduction in *Jansen* (Hrsg.), The New Economy and Beyond (http://www.stern.nyu.edu/networks/Economides_Competition_Policy.pdf – abgerufen am 30. Dezember 2015), 98.

- wird Software öfter und umfassender getestet und damit ausgereifter,
- profitiert ein Kunde von Weiterentwicklungen, die für andere Kunden vorgenommen wurden,
- können die Kunden ihre Erfahrungen mit der Software untereinander austauschen,
- kann ein Datenaustausch mit anderen Kunden ohne die Entwicklung von Schnittstellen und unter Verwendung von Standard-Dokumentenformaten erfolgen,
- stehen den Kunden eine immer größere Anzahl kompatibler bzw komplementärer Produkte (z.B. Addons, Zusatzmodule) zur Verfügung;
- steigt die Anzahl an IT-Dienstleistern, welche Customizing und Implementierung der Software anbieten.

Alle miteinander kompatiblen Installationen einer Software bei den Kunden bilden in dieser Hinsicht ein Netz, selbst wenn die Systeme nicht verbunden sind; man spricht diesfalls von einem «**virtuellen Netz**».[4]

Diese Netzeffekte können bewirken, dass sich technisch höherwertige Produkte wie z.B. Software von Startups nicht durchsetzen: Der einzelne Kunde würde zwar von einem Wechsel zum Produkt des neuen Anbieters profitieren, jedoch nur, wenn andere Kunden diesem Beispiel folgen – bleiben die anderen Kunden beim alten Produkt, entstehen beim Produkt des Startups die oben dargestellten Netzeffekte nicht und werden vom höheren Nutzen durch die verbesserten Funktionalitäten des neuen Produkts nicht kompensiert. Dies führt dann dazu, dass die meisten Kunden beim alten Produkt bleiben. Dieser Effekt wird als **Pinguin-Effekt** bezeichnet: Die hungrigen Pinguine stehen am Ufer und – da sie die Räuber im Wasser fürchten – hoffen sie, dass die anderen Pinguine zuerst ins Wasser springen, um das Risiko, gefressen zu werden, abschätzen zu können. Sobald einige Pinguine den Sprung gewagt haben, ist die Gefahr für die anderen Pinguine reduziert und die Trittbrettfahrer-Pinguine folgen.[5]

Starke Netzeffekte eines Produkts eines dominanten Software-Herstellers können bereits bei der Entscheidung über die Initialbeschaffung einer Software eine Lock-In-Situation erzeugen.

2.2. Skaleneffekte

Von Skaleneffekten («**economies of scale**») spricht man, wenn bei steigender Produktionsmenge die für die Herstellung der einzelnen Waren erforderlichen Stückkosten sinken. Bei Software sind diese Skaleneffekte sehr stark: Beinahe die gesamten Kosten fallen bei der «Produktion des ersten Stücks» – der Softwareentwicklung – an. Die Kosten für die folgenden Kopien sind aufgrund der geringfügigen Kosten der digitalen Vervielfältigung vernachlässigbar. Ein im Softwarebereich marktführendes Unternehmen gewinnt mit zunehmender Dominanz immer größeren Spielraum gegenüber der Konkurrenz und hat die Möglichkeit, diese Spielräume als Gewinne zu realisieren oder zum preislichen Unterbieten der Konkurrenten einzusetzen.[6]

Diese Skaleneffekte können bei der Entscheidung über die Initialbeschaffung einer Software eine maßgebliche Rolle spielen.

4 Economides, Competition Policy in Network Industries: An Introduction in *Jansen* (Hrsg.), The New Economy and Beyond, 96.
5 Buxmann et al., The Software Industry (2013), 24, verweisend auf Farrell and Saloner (1987).
6 Blaha, Trusted Computing auf dem Prüfstand des kartellrechtlichen Missbrauchsverbotes (2006), 80.

2.3. Kompatibilität

Die **Kompatibilität zu bestehenden Systemen** ist ein entscheidender Faktor für die Beschaffung von Software. Marktführer haben dabei tendenziell ein Interesse daran, Schnittstellenspezifikationen geheim zu halten, um Konkurrenten daran zu hindern, kompatible Produkte anzubieten.[7]

Die Frage der Kompatibilität sollte sowohl bei der Initialbeschaffung als auch bei der Entscheidung über einen Wechsel auf eine andere Software eine wichtige Rolle spielen und insbesondere bei der Ermittlung der Kosten berücksichtigt werden. Software, die auf proprietäre, mit anderen Systemen inkompatible Datenformate setzt, kann gegenüber Software, die auf Basis offener Standards arbeitet, im Bereich der Folgekosten erhebliche Mehraufwände verursachen.

2.4. Wechselkosten

Der Wechsel zu einer anderen Software erzeugt i.d.R. **hohe Wechselkosten**. Dies trifft insbesondere für Software, die eng mit den Geschäftsprozessen des Kunden zusammenhängt und diese formt, wie z.B. ERP-Software zu. Ein Wechsel zu einer anderen Software würde auch organisatorische Änderungen erforderlich machen, die zusätzliche Kosten nach sich ziehen. Eine starke Integration der bestehenden Software in das Unternehmen des Kunden und eine weitgehende Anpassung der Software an die speziellen Bedürfnisse des Unternehmens erhöhen die Wechselkosten weiter.[8]

Zu berücksichtigen sind beim Wechsel insbesondere die **Kosten der Datenübernahme** in das neue System. Hier ist insbesondere zu beachten, dass sich aus dem österreichischen Zivil- und Datenschutzrecht keine gesetzliche Verpflichtung des Dienstleisters zur Rückgabe von Daten in einem bestimmten vom Vertragspartner gewünschten Format ergibt und die Rückgabe in einem üblichen Datenformat wie pdf oder Textdateien zulässig ist.[9]

Hinzu kommt, dass im Falle eines Wechsels die in das Altsystem geflossenen Investitionen (Lizenzkosten und Kosten für Installation, Konfiguration, Customizing, Schulung) **verlorene Kosten** («sunk costs») darstellen.

Diese Kosten müssen durch entsprechend starke Vorteile wettgemacht werden, wenn ein Kunde die Software bzw das System wechseln soll. Da dies sehr schwierig ist, spricht man von **Pfadabhängigkeit**, also von der praktischen Unmöglichkeit, eine einmal getroffene Systementscheidung später zu revidieren (auch als «Lock-in» bezeichnet). Ein Kunde ist dann in einer Pfadabhängigkeit gefangen, wenn die Wechselkosten den Mehrwert des Alternativsystems übersteigen, was zur Folge hat, dass er von einem Wechsel Abstand nimmt. Die dadurch erreichte Kundenbindung kann der Hersteller über viele Generationen von Nachfolgeprodukten halten, wenn er diese Nachfolgeprodukte zur alten Software abwärtskompatibel hält.[10]

3. Interessensausgleich durch das Software-Urheberrecht

Nunmehr wird untersucht, inwieweit das Software-Urheberrecht im Hinblick auf einen möglichen Customer Lock-In-Instrumente für einen Interessenausgleich bereit stellt.

---

7 Buxmann et al., The Software Industry (2013), 28.

8 Buxmann et al., The Software Industry (2013), 28.

9 OGH 15. April 2010, 6 Ob 40/10s (Outsourcing der Personalverrechnung) = jusIT 2010, 136 (m. Anm. Staudegger/Thiele).

10 Blaha, Trusted Computing auf dem Prüfstand des kartellrechtlichen Missbrauchsverbotes (2006), 83 m.w.N.

3.1. Geschützte Ausdrucksformen von Software und zustimmungsbedürftige Handlungen

Der urheberrechtliche Schutz von Software beschränkt sich auf alle ihre **Ausdrucksformen** (Sourcecode, Maschinencode, Programmdokumentation). Nicht urheberrechtlich geschützt sind die Ideen und Grundsätze, die der Software und ihren Schnittstellen zugrunde liegen.[11] Dasselbe gilt für die Ideen und Grundsätze, die der Logik, den Algorithmen und den Programmsprachen zugrunde liegen.[12] Weiter sind auch die Funktionalität der Software, die Nutzung der im Rahmen eines Programmes vorgesehenen Programmiersprachen und die Datenformate nicht urheberrechtlich geschützt.[13] Auch eine graphische Benutzeroberfläche stellt keine Ausdrucksform von Software dar und ist daher nicht als Software (allenfalls jedoch als eigenes Werk) urheberrechtlich geschützt.[14]

Dem Urheber vorbehalten sind nach der Software-Richtlinie die **Vervielfältigung, Übersetzung, Bearbeitung sowie die öffentliche Verbreitung** der Software.[15]

3.2. Ausnahmen

Nicht der Zustimmung des Urhebers bedürfen

- Vervielfältigung, Übersetzung und Bearbeitung, soweit sie für die **bestimmungsgemäße Benutzung der Software** einschließlich der Fehlerberichtigung[16] erforderlich sind,[17]
- die für die Benutzung erforderliche Erstellung einer **Sicherungskopie**,[18]
- das **Beobachten, Untersuchen oder Testen des Funktionierens** der Software, um die einem Programmelement zugrundeliegenden Ideen und Grundsätze zu ermitteln («Reverse Engineering»), soweit sie durch berechtigte Handlungen zum Laden, Anzeigen, Ablaufen, Übertragen oder Speichern des Programms erfolgen,[19]
- das **Dekompilieren** der Software, sofern es für das Erlangen der zur Herstellung der Interoperabilität mit anderer Software erforderlichen und nicht bereits ohne weiteres zugänglich gemachten Informationen unerlässlich ist und nicht über den notwendige Umfang hinausgeht.[20]

3.3. Bewertung

Die oben angeführten Ausnahmen von den Ausschließlichkeitsrechten des Urhebers sind nur in einem sehr engen Rahmen geeignet, Lock-In-Situationen zu vermeiden bzw aufzulösen:

Das Recht zur Vornahme der **für die bestimmungsgemäße Benutzung erforderlichen Handlungen** samt dem Recht zur Erstellung einer Sicherungskopie sichert lediglich die Möglichkeiten des rechtmäßigen Erwerbers zur tatsächlichen Nutzung der Software ab. Darüber hinausgehende Interessen, wie ein wirtschaftlich

11 Art. 1 Abs. 2 RL 2009/24/EG.
12 Erwägungsgrund 11 RL 2009/24/EG.
13 EuGH 2. Mai 2012, Rs C-406/10 «SAS Institute» = jusIT 2012, 97 (m. Anm. Staudegger).
14 EuGH 22. Dezember 2010, Rs C-393/09 «BSA/cz Kulturministerium» = jusIT 2011, 44 (m. Anm. Staudegger und Thiele).
15 Art. 4 Abs. 1 RL 2009/24/EG.
16 Das österreichische Urheberrecht geht darüber hinaus und erlaubt auch die Anpassung der Software an die eigenen Bedürfnisse. Die Richtlinienkonformität dieser Regelung ist jedoch fraglich – siehe Blocher in *Jahnel/Mader/Staudegger*, IT-Recht[3] (2012), 236 f.
17 Art. 5 Abs. 1 RL 2009/24/EG.
18 Art. 5 Abs. 2 RL 2009/24/EG.
19 Art. 5 Abs. 3 RL 2009/24/EG.
20 Art. 6 Abs. 3 RL 2009/24/EG.

vertretbarer Umstieg auf eine andere Software, sind davon nicht umfasst.

Das Recht zum **Beobachten, Untersuchen und Testen** des Funktionierens der Software hat den Fokus weniger auf der Vermeidung von Lock-In-Situationen, sondern mehr auf der Absicherung des Grundsatzes, dass nur die Ausdrucksformen von Software, nicht aber die zugrunde liegenden Ideen geschützt sind, und dem Ermöglichen der Entwicklung ähnlicher Software. Die Nutzung dieses Rechts ist kaum geeignet, den Aufwand einer Migration zu einer neuen Software in Grenzen zu halten.

Die **Dekompilierung** kann infolge einer aufgrund von Inkompatibilitäten der Software vorliegenden Lock-In-Situation zulässig sein. Die Dekompilierung ist jedoch nur in den engen oben angeführten Grenzen rechtlich zulässig. Weiters kann Dekompilierung mit technischen Maßnahmen wie Code-Obfuskation oder Verschlüsselung des Codes erschwert oder ganz verhindert werden und damit hohe Kosten mit sich bringen.

Als Ergebnis lässt sich festhalten, dass das Urheberrecht lediglich in Zusammenhang mit der Problematik der Interoperabilität ein (sehr eng begrenztes) Mittel in die Hand gibt, eine Lock-In-Situation zu vermeiden. Tatsächlich fällt der rechtliche Umgang mit Geschäftspraktiken, die zu Lock-In-Situationen führen, in den Bereich der **kartellrechtlichen Tatbestände** des Ausbeutungs- und Behinderungsmissbrauchs.[21]

4. Vertragliche Instrumente

Da das Urheberrecht kaum Abhilfe schaffen kann, ist der Kunde selber in der Verantwortung, mögliche Lock-In-Situationen zu antizipieren, die Risiken zu bewerten und bei der Softwarebeschaffung und Vertragsgestaltung zu berücksichtigen:

4.1. Einbeziehung der maßgeblichen Faktoren

In die Bewertung der Vor- und Nachteile der angebotenen Lösungen sollten daher folgende Faktoren einbezogen werden:

- Arbeitet die angebotene Software mit offenen oder proprietären Datenformaten?
- Arbeitet die angebotene Software mit offenen oder proprietären Schnittstellen?
- Enthält die Software Programmsperren oder technische Schutzmaßnahmen?
- Welche Lizenzierungsmodelle bietet der Hersteller der Software an?
- Welche Politik hat der Hersteller der angebotenen Software im Hinblick auf diese Aspekte in der Vergangenheit verfolgt?
- Weist die angebotene Software eine für den Auftraggeber auch tatsächlich nutzbare Funktion für den Export der Daten in strukturierter Form auf?
- Wie viele weitere Unternehmen (z.B. Reseller, zertifizierte Partner des Herstellers der Software) mit entsprechend qualifizierten Entwicklern, welche die angebotene Software warten und weiterentwickeln können, gibt es außer dem potenziellen Auftragnehmer noch?
- In welchen Umfang lässt sich die angebotene Software bei künftige Änderungen der organisatorischen oder gesetzlichen Rahmenbedingungen anpassen?
- Wie lange wird die angebotene Standardsoftware vom Hersteller noch unterstützt (Patches, Updates)?

[21] Siehe grundsätzlich Koppensteiner, Markt, Wettbewerb und Vertrag, JBl 2015, 137 und zur IT Thalmann, Der Fall Microsoft als Exerzierfeld eines «more economic approach»?, wbl 2008, 153.

- Wie wird vorgegangen, wenn sie nicht mehr unterstützt wird?
- Welche Exit-Szenarien gibt es für eine Migration aus dem angebotenen System? Welche technischen Schwierigkeiten sind in diesen Fällen zu erwarten?
- Welche Erfahrungen haben andere Kunden bei der Migration aus dem angebotenen System gemacht?

Wesentlicher Teil dieser Arbeit ist die ernsthafte Ermittlung der **Lebenszykluskosten** («Total Cost of Ownership») der angebotenen Lösungen von der Konzeptionsphase bis zur Migration aus dem System. Dabei ist man gut beraten, sich beim Preisvergleich nicht durch niedrige Initialkosten blenden zu lassen, da sich die Folgekosten im Falle eines Lock-In wesentlich stärker auswirken können.

4.2. Vertragsgestaltung

Bei der Vertragsgestaltung sollten im Hinblick auf die Vermeidung einer künftigen Lock-In-Situation folgende Punkte berücksichtigt werden:

- Nach welchen technischen Normen erfolgt die Entwicklung?
- Welche überprüfbaren Anforderungen können im Hinblick auf die Wartungsfreundlichkeit der Software definiert werden?
- Welche überprüfbaren Software-Qualitätskriterien können definiert werden?
- Wie überprüft der Auftraggeber die Einhaltung dieser Kriterien und Anforderungen?
- In welchem Umfang stellt der Auftragnehmer eine Dokumentation der Software zur Verfügung?
- Ist der Auftragnehmer mit der Zurverfügungstellung bzw Hinterlegung des Sourcecodes bereit?
- Welche Software- bzw Systemteile sind vom zur Verfügung gestellten bzw hinterlegten Sourcecode umfasst?
- Für welche Software- bzw Systemteile kann kein Sourcecode zur Verfügung gestellt bzw hinterlegt werden (z.B. weil es sich um Standardsoftware eines Dritten handelt)?
- Ist mit dem zur Verfügung gestellten bzw hinterlegten Sourcecode eine Wartung bzw Weiterentwicklung überhaupt möglich?
- Unter welchen Voraussetzungen und Bedingungen ist der Auftraggeber berechtigt, den Sourcecode zur Wartung und Weiterentwicklung zu verwenden bzw einen Dritten damit zu beauftragen?
- Wie überprüft der Auftraggeber einen vom Auftragnehmer gelieferten bzw bei einem Dritten zu hinterlegenden Sourcecode?
- Welche Kosten sind mit der Prüfung und Hinterlegung des Sourcecodes verbunden? Sind diese im Projektbudget vorgesehen?
- Welche für die Wartung und Weiterentwicklung erforderlichen Rechte werden dem Auftraggeber übertragen?
- Wie wird sichergestellt, dass der Auftraggeber diese Rechte auch tatsächlich ausüben kann?
- Welche konkreten und verbindlichen Mitwirkungsleistungen des Auftragnehmers können für den Fall einer Migration zu einer anderen Software festgelegt werden?

Bei der Vertragsgestaltung ist es wichtig, dass die vertraglichen und technischen Regelungen (Leistungsbeschreibung, Lastenheft) wie Zahnräder ineinandergreifen. Dies setzt eine enge Zusammenarbeit zwischen Technikern und Juristen voraus. Wie die beiden eingangs beschriebenen Beispiele zeigen, sind Vertragsklauseln, die für sich isoliert stehen und nicht auf die technischen Realitäten abgestimmt sind, nutzlos.

5. Resümee

Die Bereitschaft, über die dargestellten Punkte zu verhandeln und verbindliche Verpflichtungen festzulegen, ist in der Regel umgekehrt proportional zur Marktmacht des Verhandlungspartners. Aber auch wenn Lizenzverträge über Standardsoftware großer Hersteller nicht verhandelbar sind, lassen sich in Verhandlungen mit mittelständischen, aber auch großen IT-Dienstleistern und Systemhäusern meist Lösungen für diese Problemstellungen finden.

Aus Kundensicht ist es jedenfalls erforderlich, die zur Vermeidung erforderlichen Weichenstellungen vor Vertragsabschluss vorzunehmen. Ein Vertagen des Themas auf später ist nicht empfehlenswert, da dann die Verhandlungsposition des Kunden wesentlich schwächer ist.

6. Literatur

Buxmann et al., The Software Industry – Economic Principles, Strategies, Perspectives, Springer Verlag, Berlin Heidelberg 2013.

Blaha, Trusted Computing auf dem Prüfstand des kartellrechtlichen Missbrauchsverbotes, Verlag Österreich, Wien 2006.

Economides, Competition Policy in Network Industries: An Introduction in *Jansen* (Hrsg.), The New Economy and Beyond – Past, Present and Future, Edward Elgar Publishing, 2006.

Jahnel/Mader/Staudegger, IT-Recht[3], Verlag Österreich, Wien 2012.

Koppensteiner, Markt, Wettbewerb und Vertrag, JBl 2015, 137.

Staudegger/Thiele, EuGH: Zum urheberrechtlichen Schutz grafischer Benutzeroberflächen, jusIT 2011, 44.

Staudegger, EuGH: Schutzgegenstand Computerprogramm, jusIT 2012, 97.

Thalmann, Der Fall Microsoft als Exerzierfeld eines «more economic approach»?, wbl 2008, 153.