SMART CONTEXTS FÜR SMART CONTRACTS – LEGAL PROGRAMMING FÜR VALIDE BLOCKCHAIN-VERTRÄGE

Peter Ebenhoch / Felix Gantner

Dr., PMP, Senior Consultant, effectas GmbH
Bundesstrasse 6, 6300 Zug, CH
peter.ebenhoch@effectas.com; http://www.lean-grc.com

DI Dr., infolex Rechtsinformatik
Bei der Kapelle 7, 3592 Röhrenbach, AT
gantner@infolex.at; http://www.infolex.at

Schlagworte: ***Blockchains, Smart Contracts, Legaltech, Legal Programming, Legal Tech, Legal Writing***

Abstract: *Blockchains wie Etherum erlauben mit Smart Contracts den Abschluss von Rechtsgeschäften. Die Wirksamkeit dieser digitalen Artefakte setzt eine Verankerung im beherbergenden Rechtssystem voraus. Dazu werden «Smart Contexts» als rechtlich verbindliche Begleitartefakte etabliert. Sie liefern den Zusatzkontext, um einen Smart Contract rechtlich wirksam zu machen. Im Sinne eines «Legal Programming» können diese mit einer integrierten Editorenumgebung erstellt werden, die auf das Literate Programming von Donald Knuth zurückgreift und kontrollierte Vokabulare sowie eine formale Notation einsetzt.*

1. Rechtssemiotische Grundlage für Smart Contracts

1.1. Das semiotische Dreieck

Auf Charles W. Morris[1] geht diejenige Version des ursprünglich von Charles Sanders Peirce[2] entwickelten semiotischen Dreiecks zurück, die die *semantische, pragmatische und syntaktische Dimension* des Zeichens und seiner Bedeutung bzw. von Information allgemein und von Rechtsinformation im Besonderen unterscheidet. Ein Zeichen (syntaktische Dimension) hat demnach eine bestimmte Bedeutung (semantische Dimension), die in einem bestimmten Anwendungskontext (pragmatische Dimension) relevant wird.

1.2. Vertragliche Rechtsgestaltung veranschaulicht am semiotischen Dreieck

Bei einer normalen Vertragsgestaltung werden in einer bestimmten Situation (Pragmatik) bestimmte rechtliche Bedeutungen und Wirkungen (Semantik) durch übereinstimmende Willenserklärungen in einem Schriftstück fixiert (Syntax) und für rechtlich verbindlich erklärt. Anschliessend wird der Vertragstext (Syntax) realisiert, d.h. im Alltag verwirklicht (Pragmatik).

---

1 Morris.
2 Peirce.

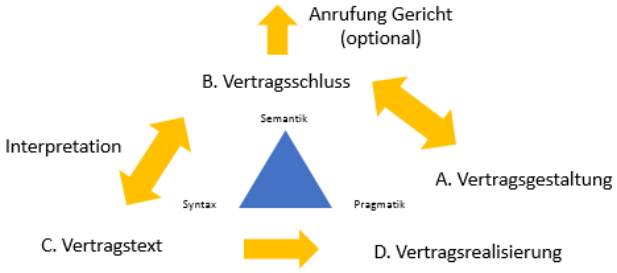

Abbildung 1: Herkömmliche Vertragsgestaltung

Irrten beide Parteien gemeinsam, so ist das unerheblich, selbst wenn der Vertragstext abweicht («falsa demonstratio non nocet»). Bei einseitigen Irrtümern oder in Streitfällen steht jederzeit wieder die semantische Ebene zur Verfügung, durch erneute Diskussion mit dem Vertragspartner, notfalls unter Anrufung einer Behörde (Gericht) und eines Verfahrens.

1.3. «Smart Contracts» im semiotischen Dreieck

Bei Smart Contracts werden in einer bestimmten Situation (Pragmatik) bestimmte rechtliche Bedeutungen und Wirkungen (Semantik) durch übereinstimmende Willenserklärungen aus der natürlichen Sprache in einer abstrakte Programmiersprache als Softwarecode fixiert (Syntax) und dieses auf einer Blockchain initialisiert. Der Programmcode wird anschließend automatisch realisiert, d.h. im Alltag verwirklicht (Pragmatik).

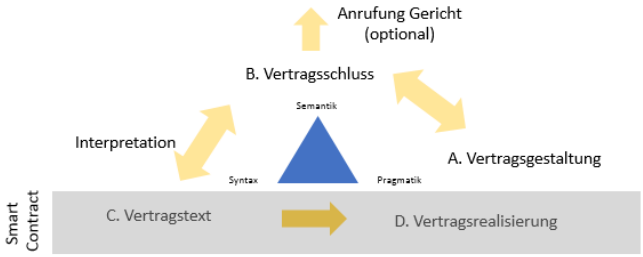

Abbildung 2: Smart Contracts

Irrten beide Parteien gemeinsam, so ist das erheblich, da der in Programmform abgefasste Vertragstext abweicht («falsa demonstratio nocet»). Bei einseitigen Irrtümern oder in Streitfällen fehlt die semantische Ebene und mangelt es an einem Verfahren zur Streitbeilegung. Selbst eine Diskussion mit dem Vertragspartner ist wegen möglicher Anonymität nicht sicher möglich. Die programmierte Vertragsform entwickelt somit ein Eigenleben, das selbst bei Einvernehmen nur mehr nach den Regeln und Mechanismen der Blockchain angepasst werden kann (Consensus, Wartezeit).

Spätestens hier wird klar, dass ein isolierter Smart Contract nur bedingt mit der Alltagsrealität zu tun hat und sich sogar gewissermassen «autistisch» von den eigentlichen Absichten der Vertragspartner entfernen kann. Isolierte Smart Contracts sind also nur beschränkt alltagstauglich. Bøgh-Anderson kann nur zugestimmt werden, wenn er schreibt: «We quickly discover, that it is only half-truth that the source code stands for the domain»[3].

[3] Bøgh-Anderson, S. 4.

1.4. Legal Programming mit Smart Contexts

Das nachfolgend vorgestellte Konzept des *Legal Programmings* und von *Smart Contexts* lässt sich folgendermassen visualisieren: Der «Smart Contract» wird Teil des «Smart Contexts» als Gesamtvertrag.

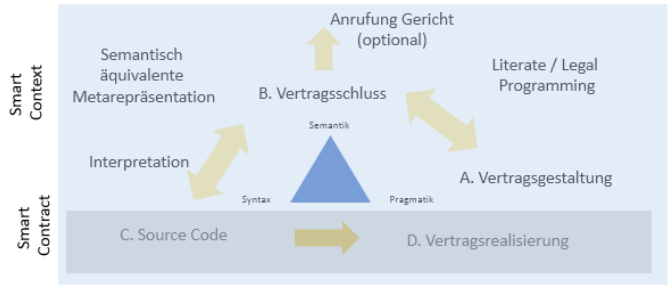

Abbildung 3: Smart Contracts einbegettet in Smart Contexts

Durch Nutzung eines kontrollierten Vokabulars kann der Programmcode in einer semantisch äquivalenten Darstellung (Metarepräsentation) dargestellt werden. Diese Metarepräsentation ist syntaktisch mit dem Sourcecode gekoppelt und bleibt als User Story oder Test Case IT-affin, ist aber für Menschen einfach nachvollziehbar als abstrakter Programmcode. Der Smart Contract erhält dadurch einen alltagstauglichen Smart Context.

2. Blockchains als Grundlage für Smart Contracts

Blockchains ermöglichen die dezentrale Speicherung von Daten, bei der durch kryptographische Methoden sichergestellt wird, dass bereits gespeicherte Daten nicht mehr verändert werden können. Änderungen des Datenbestandes sind nur durch Hinzufügen neuer Datensätze (Blöcke) möglich. Der alte Datenbestand bleibt einsehbar und alle Änderungen sind dauerhaft nachvollziehbar.

Daten werden in einer Blockchain in einer schreibgeschützten und kryptografisch signierten Dateneinheit, dem Block, gespeichert. Ein Block kann beliebige Daten enthalten. Dazu kann auch Programmcode zählen.

Ein Blockchain-System selbst kann ─ je nach Verbreitung und Nutzerkreis – als ein bis zu weltweit verteiltes Computersystem angesehen werden. Jeder einzelne Knoten, also jeder Rechner, wird dabei als Teil der Blockchain synchron mit allen anderen Knoten gehalten. Die Blockchain kann damit als ein einziger, verteilter virtueller Computer angesehen werden.

Ist in einem Block Programm-Code gespeichert, so kann der virtuelle Computer angewiesen werden, diesen Programm-Code sofort oder zu einem bestimmten Zeitpunkt und auch abhängig von Bedingungen auszuführen. Solche Programme werden als Smart[4] Contract bezeichnet.

In Blockchain-System Ethereum[5] werden Smart Contracts in der Programmiersprache Solidity[6] geschrieben und können dann auf allen Rechnern des Ethereum-Netzwerks, die gemeinsam die «Ethereum Virtual Machine» (EVM)[7] bilden, ausgeführt werden.

---

4 Die Bezeichnung «smart» ist im Zusammenhang mit Blockchains nicht als «schlau», «intelligent» oder ähnliches zu interpretieren, sondern wird in diesem Kontext verwendet, wenn eindeutig identifizierbare Objekte bearbeitet werden. Vgl. Swan, S viii (Hervorhebung im Original): «In this system, all property could become *smart property*; this is the notion of encoding every asset to the blockchain with a unique identifier such that the asset can be tracked, controlled, and exchanged (bought or sold) on the blockchain.»

5 https://www.ethereum.org (alle Websites zuletzt besucht im Januar 2018).

6 http://solidity.readthedocs.io/en/develop.

7 Dannen, S. 47 ff.

Smart Contracts sind daher (mitunter sehr komplexe) Programme, die, sobald sie Teil der Blockchain sind, nicht mehr verändert und im Netzwerk ausgeführt werden. Typische Anwendungsfälle sind Transaktionen, die mit Kryptowährungen in Verbindung stehen, also z.B. Überweisungen bzw. Änderungen von Kontoständen, die durch das Programm vorgenommen wenden.

Auch wenn die Bezeichnung «Smart Contract» anderes suggeriert, werden die Programme alleine in den seltensten Fällen[8] als Vertrag im juristischen Sinn einzuordnen sein. Natürlich könnten Vertragsparteien Verträge auch exklusiv in einer Programmiersprache formulieren und im Einvernehmen das Programm bei Vertragsschluss auch gleich ausführen lassen. Dies wird allerdings alleine aus Gründen der mangelnden Lesbarkeit und Verständlichkeit[9] in den seltensten Fällen sinnvoll und möglich sein.

In Zukunft wird es jedoch immer häufiger vorkommen, dass Verträge um ein oder mehrere Programme erweitert werden. Diese Softwareartefakte sind dann Teil des Vertragstextes. Bei Vertragsschluss kann dann sowohl der Vertragstext, als auch der ausführbare Programmcode gemeinsam in einem neuen Block der Blockchain gespeichert werden.

Der Code ist dann ein Teil der Umsetzung des Vertrags und der Vertrag die Grundlage, aber aus Sicht der Softwareentwicklung zusätzlich sowohl Pflichtenheft, als auch Dokumentation des Programms.

Es entwickelt sich damit eine neue Art juristischer Artefakte, eine neue juristische Dokumentform, als Kombination aus «klassischem» Rechtstext, in diesem Fall juristischen Vertragsformulierungen, und in einer Programmiersprache formuliertem Quelltext von Software. Aus juristischer Sicht ist der «Smart Contract», also der Programmcode, Teil des Vertrags und ein Mittel der Vertragserfüllung. Aus Sicht der Software-Entwicklung ist der «klassische» Vertragstext der Kontext, in dem das Programm entwickelt wird. Er wird im Folgenden daher als «Smart Context» bezeichnet.

Der Vertrag ist die Kombination von «Smart Context» und «Smart Contract».

Die folgenden Ausführungen entwickeln die Methode des «Legal Programming», mit der sichergestellt werden soll, dass valide, d. h. «gerichtsfeste»[10] Blockchain-Verträge verfasst und programmiert werden.

3. Legal Programming

Ausgangspunkt für Legal Programming ist das von Donald E. Knuth entwickelte Konzept des «Literate Programming». Programmcode wird dabei als eine Form von Literatur angesehen.[11] Im Programmcode werden Kommentare und Dokumentationselemente mit besonderen Textauszeichnungen versehen und aus ein und derselben Quelle werden die Dokumentation und die das ausführbare Programm erzeugt[12].

Beim Entwurf von Verträgen, die Smart Context und Smart Contract enthalten, kann von einer ähnlichen Problemstellung ausgegangen werden:

- Programmcode und Vertragstexte können beide als Form der Literatur angesehen werden.
- Zwischen den einzelnen Textelementen bestehen enge inhaltliche Verbindungen.

---

8 Dülpers; Kargl.

9 Ab einer gewissen Komplexität ist jede Software fehlerbehaftet und nicht in mehr in allen Aspekten überblickbar, weshalb sich bei Fehlverhalten des Programms immer die Frage stellen wird, ob dies Teil der Vereinbarung ist. Bei Smart Contracts wird diese Problematik noch verschärft, da Fehler im Code nicht einfach behoben werden können, da die Blöcke einer blockchain nicht verändert werden können und verteilt ausgeführter Code nicht an einer zentralen Stelle ausgeführt wird. Ein einfaches Update ist damit nicht möglich. Dies zeigte auch der Fall der Blockchain «The DAO» (vgl. dazu Greenspan; Abegg).

10 Vgl. Adrian, S. 114: «Man könnte überspitzt sagen, dass die Verträge dann gerichtsfest sind, wenn der Vertragstext dem Text strukturell möglichst ähnlich ist, der in dem Nachschlagewerk enthalten ist, das der Richter benutzt, wenn er über die juristische Qualität dieses Vertrages zu entscheiden hat.»

11 Knuth, S. 1 (Hervorhebung im Original): «I believe that the time is ripe for significantly better documentation of programs, and that we best can achieve this by considering programs to be *works of literature*.»

12 Knuth, S. 2.

– Das Endergebnis sollen zwei unterschiedliche Texte sein, wobei einer davon die Grundlage für ausführbaren Programmcode ist.

Ein Unterschied ist die Qualifikation bzw. das inhaltliche Hauptinteresse des/der Autoren. Während bei «Literate Programming» Software-Entwickler die Texte erstellen und das Hauptinteresse auf dem ausführbaren Programm als zu erzeugendem Endprodukt liegt, ist es beim «Legal Programming» umgekehrt. Der juristische Vertragstext und die darauf bezogene Einigung der Parteien, also der Smart Context, ist Gegenstand des juristischen Interesses, der Smart Contract ein notwendiges Nebenprodukt. Ähnlich wie die Dokumentation von Software, die vom Programmcode abhängig ist, ergibt sich der Smart Contract aus dem Smart Context.

Selbst wenn Juristen auch im Bereich der Software-Entwicklung ausgebildet[13] wären, würde sich an dieser Schwerpunktsetzung und Gewichtung in der Bedeutung der Textteile nichts ändern.

Im Folgenden wird in mehreren Schritten an Hand von bearbeiteten und erzeugten Texten beispielhaft die Methode des Legal Programming beschrieben. Dabei wird an zwei kurzen Beispieltexten dargestellt, die bereits das Endprodukt der Entwicklung von Smart Content und Smart Contract sind. Durch Struktur- und Textanalyse werden beim Legal Programming Texte gefunden, aus denen der Smart Contract automatisiert erzeugt werden kann.

3.1. Minimale Beispieltexte

Bei den Beispieltexten sind weder der juristische Inhalt, noch der Programmcode von zentraler Bedeutung. Vielmehr geht es um strukturelle Zusammenhänge und die Möglichkeiten der automatisierten Texterzeugung analog zum Konzept des «Literate Programming».

3.1.1. Text des Smart Context

Der folgende minimale Text eines Kaufvertrags ist der Beispieltext für den Smart Context:

```
Kaufvertrag
zwischen XXX (nachfolgend Verkäufer) und YYY (nachfolgend Käufer).
Der Verkäufer veräußert an den Käufer ZZZ (nachfolgend Kaufgegenstand).
Der Kaufpreis beträgt 0,10 ETH (Ether) und wird mittels smart contract
bezahlt, wenn der Kaufgegenstand übergeben wurde.
```

3.1.2. Text des Smart Contract

Der folgende Anfang eines Solidity-Programmcodes ist der Beispieltext für den Smart Contract:[14]

```
pragma solidity ^0.4.11;
contract Purchase {
    uint public value;
    address public seller;
    address public buyer;

    function Purchase() public payable {
        seller = msg.sender;
        buyer = msg.buyer;
        value = msg.value;
…
```

[13] Vgl. Siegel: «I think in five years most law schools will be teaching, probably requiring, coding skills. Thinking like a lawyer and thinking like a programmer aren't that different.»

[14] Das Beispiel ist angelehnt an die unter http://solidity.readthedocs.io/en/develop/common-patterns.html beschriebenen Codebeispiele für die Programmiersprache Solidity des Ethereum-Projekts.

3.2. Schritt 1: Vertrags-Parameter

Typisch für standardisierbare juristische Texte, die teilweise auch formularähnlichen Charakter haben, ist die Möglichkeit, Parameter zu identifizieren. Während die juristischen Textteile und -inhalte bei ähnlichen Texten einander ähneln, werden die Texte durch die den Parametern zugewiesenen Werte individualisiert.[15]

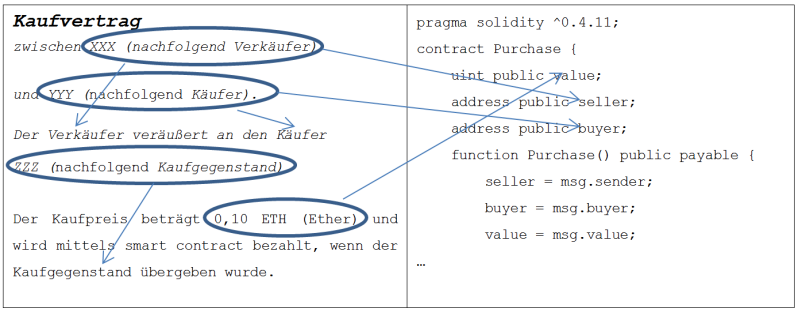

Tabelle 1: Parameter in Smart Context und in Smart Contract

Typisch für Parameter ist, dass sie bereits im Text auch als solche verwendet werden («nachfolgend Verkäufer»), wenn der Text als Vorlage wiederholt genutzt wird.

Parameter im Smart Contract, für die von der Blockchain an das Programm bei der Programmausführung Werte übergeben werden, sind jedenfalls auch im Smart Context zu finden. Parameter des Smart Context müssen nicht in jedem Smart Contract enthalten sein.

3.3. Schritt 2: Markup

Nach Identifikation der Parameter ergibt sich die interne Textdarstellung, bei der automatisierten Generierung des endgültigen Smart Context und Smart Contract, also jener Texte, die in der Blockchain gespeichert werden, verwendet werden.

Smart Context	Smart Contract
Kaufvertrag *zwischen [XXX](Verkäufer)|seller| und [YYY](Käufer)|buyer|.* *Der #Verkäufer veräußert an den #Käufer [ZZZ](Kaufgegenstand).* Der #Kaufpreis beträgt [0,10 ETH](Kaufpreis)|value| und wird mittels smart contract bezahlt, wenn der #Kaufgegenstand übergeben wurde.	pragma solidity ^0.4.11; contract Purchase { uint public value; address public seller; address public buyer; function Purchase() public payable { seller = #sender; buyer = #buyer; value = #value; …

Tabelle 2: Parameter in Smart Context und in Smart Contract unter Verwendung von Markup

Es wird dabei eine Textauszeichnung, die an Markdown angelehnt ist verwendet. Textauszeichnungen wie Überschriften, … werden nicht dargestellt.

[15] Hazard, James/Haapio, Helena, Wise Contracts: Smart Contracts that Work for People and Machines, S. 425f.

3.4. Schritt 3: notwendige Abstraktion

Der in Schritt 2 verwendete Text hat einige wesentliche Nachteile:

- Programmcode ist noch immer im bearbeiteten Text enthalten.
- Die Texte wurden noch weniger verständlich und bearbeitbar, als in den Ausgangstexten.
- Programmierkenntnisse wären zum Nachbearbeiten der Texte notwendig.
- Das Verhalten des Smart Contracts ist für die Vertragsparteien nicht vollständig nachvollziehbar.

Smart Context	Smart Contract
Kaufvertrag *zwischen [XXX](Verkäufer)|seller| und [YYY] (Käufer)|buyer|.* *Der #Verkäufer veräußert an den #Käufer [ZZZ] (Kaufgegenstand).* Der #Kaufpreis beträgt [0,10 ETH](Kaufpreis)|value| und wird mittels smart contract bezahlt, wenn der #Kaufgegenstand übergeben wurde.	Als #Verkäufer bin Besitzer des Smart contracts. Wenn #Kaufgegenstand übergeben wurde, dann überweist #Käufer #Kaufpreis an #Verkäufer. Wenn #Kaufgegenstand nicht übergeben wurde dann überweist #Käufer #Kaufpreis nicht an #Verkäufer.

Tabelle 3: Definition des Smart Contract

Die Lösung dieser Probleme kann durch die Methoden des Software-Engineering gefunden werden. Insbesondere die Ansätze des test-driven development (TDD) und des behavior-driven development (BDD)[16] weisen den Weg zur notwendigen sprachlichen und inhaltlichen Abstraktion. Dabei kann für die Definition des Smart Contracts nur aus einem vorgegebenen Wortvorrat ausgewählt werden, was die automatisierte Erzeugung des Programmcodes ermöglicht.

Wenn das Verhalten des Smart Contracts durch Beschreibungen wie

```
Wenn #Kaufgegenstand übergeben wurde, dann überweist #Käufer #Kaufpreis
an #Verkäufer.
```

definiert wird, dann sind sowohl die Parameter-Werte, als auch das Verhalten des Smart-Contracts definiert.

Wie im TDD üblich, werden auch Beschreibungen des Verhaltens bei nicht erwünschten Fällen bzw. nicht erfüllten Bedingungen formuliert:

```
Wenn #Kaufgegenstand übergeben wurde, dann überweist #Käufer #Kaufpreis
an #Verkäufer.
```

Dies ist deshalb notwendig, weil der Smart-Contract selbständig in der Blockchain ausgeführt wird und ein Eingreifen bei Fehlerhaftem Programmcode nicht möglich ist. Eine möglichst umfangreiche Absicherung gegen Fehlverhalten ist daher notwendig.

4. Legal Programming: der Editor als Simulator des Smart Contracts

In einer modernen Software, die Legal Programming unterstützt, ist die Benutzeroberfläche, von besonderer Bedeutung. Mit den oben beschriebenen Datenstrukturen und Markup-Texten werden vor die Benutzer nicht behelligt.

Vielmehr kann im Legal Programming-Editor zusätzlich zur üblichen Textverarbeitung eine Simulation des Verhaltens des Smart Contracts durchgeführt werden. Diese kann die Generierung von Programmcode für die Blockchain und einem Probelauf in einer Simulationsumgebung sein.

[16] Vgl. z.B. https://cucumber.io.

Da aber Dank der für BDD typischen standardisierten Beschreibung des Systemverhaltens eine Ausgabe und Konvertierung der Beschreibung in unterschiedliche Datenformate oder Medien möglich ist, wäre eine Darstellung als Grafiken oder Animationen möglich. Diese könnten dann sogar als zusätzliche Dokumentation in den Smart Context übernommen werden.

5. Literatur

Abegg, Lukas, Code is law? Not Quite Yet, coindesk, https://www.coindesk.com/code-is-law-not-quite-yet/, 27. August 2016.

Adrian, Axel, Der Richterautomat ist möglich – Semantik ist nur eine Illusion, Rechtstheorie 2017, S. 77–121.

Bøgh-Anderson, Peter, A Theory of Computer Semiotics: Semiotic Approaches to Construction and Assessment of Computer Systems, Cambridge 2008.

Dannen, Chris, Introducing Ethereum and Solidity, Apress, New York 2017.

Dülpers, Christian, Smart contracts sind weder smart noch contracts, Legal Tribune Online, 2017, S. 21–24.

Erbguth, Jörn, Lösung Blockchain-basierter Konflikte. In: Schweighofer, Erich/Kummer, Franz/Hötzendorfer, Walter/Sorge, Christoph (Hrsg.), Trends und Communities in der Rechtsinformatik. Tagungsband des 20. Internationalen Rechtsinformatik Symposions IRIS 2017, books@ocg.at, Wien 2017, S. 155–160.

Fill, Hans-Georg, Metamodelling as an interface between syntax and semantics. In: Glück, Beate/Lachmayer, Friedrich/Schefbeck, Günther/Schweighofer, Erich (Hrsg.), Elektronische Schnittstellen in der Staatsorganisation. Festschrift zum 60. Geburtstag von Dr. Josef Souhrada, Österreichische Computer Gesellschaft, Wien 2015, S. 43–50.

Greenspan, Gideon, Smart contracts and the DAO implosion, MultiChain, https://www.multichain.com/blog/2016/06/smart-contracts-the-dao-implosion/, 22. Juni 2016.

Haapio, Helena, Designing Readable Contracts: Goodbye to Legal Writing – Welcome to Information Design and Visualization. In: Schweighofer, Erich/Kummer, Franz/Hötzendorfer, Walter (Hrsg.), Abstraktion und Applikation. Tagungsband des 16. Internationalen Rechtsinformatik Symposions IRIS 2013, books@ocg.at, Wien 2017, S. 445–452.

Hazard, James/Haapio, Helena, Wise Contracts: Smart Contracts that Work for People and Machines. In: Schweighofer, Erich/Kummer, Franz/Hötzendorfer, Walter/Sorge, Christoph (Hrsg.), Trends und Communities in der Rechtsinformatik. Tagungsband des 20. Internationalen Rechtsinformatik Symposions IRIS 2017, books@ocg.at, Wien 2017, S. 425–432.

Kahlig, Wolfgang, Ist Rechtsklarheit überhaupt erwünscht? In: Glück, Beate/Lachmayer, Friedrich/Schefbeck, Günther/Schweighofer, Erich (Hrsg.), Elektronische Schnittstellen in der Staatsorganisation. Festschrift zum 60. Geburtstag von Dr. Josef Souhrada, Österreichische Computer Gesellschaft, Wien 2015, S. 109–117.

Kahlig, Wolfgang/Kahlig, Eleonora, Rechtsvisualisierung – Viribus Unitis – mit C.O.N.T.E.N.T. In: Schweighofer, Erich/Kummer, Franz/Hötzendorfer, Walter (Hrsg.), Kooperation. Tagungsband des 18. Internationalen Rechtsinformatik Symposions IRIS 2015, books@ocg.at, Wien 2015, S. 425–442.

Kargl, Rolf-Dieter, Gesetzeslücken?, read_it, 02/2017, S. 10.

Knuth, Donald e., Literate Programming, http://www.literateprogramming.com/knuthweb.pdf, 1983.

Lenk, Klaus, Die neuen Instrumente der weltweiten digitalen Governance, Verwaltung und Management, 2016, S. 227–240.

Mielke, Bettina/Walser Kessel, Caroline/Wolff, Christian, 20 Jahre Rechtsvisualisierung – Bestandsaufnahme und Storytelling. In: Schweighofer, Erich/Kummer, Franz/Hötzendorfer, Walter/Sorge, Christoph (Hrsg.), Trends und Communities in der Rechtsinformatik. Tagungsband des 20. Internationalen Rechtsinformatik Symposions IRIS 2017, books@ocg.at, Wien 2017, S. 377–386.

Morris, W. Charles, Foundations of the theory of signs Chicago, Ill: Univ. Chicago Press. 1938.

Siegel, David, It's Time for Smart Law, Medium, https://medium.com/@pullnews/its-time-for-smart-law-f218598dee92, 2. Mai 2017.

Swan, Melanie, Blockchain, O'Reilly, Sebastopol 2015.