Selfaware Monopoly

Martin Fischer, Pascal Crenzin, Tobias Knöschke
Digital Engineering Fakultät, Hasso-Plattner-Institut | Universität Potsdam

Abstract

In dieser Ausarbeitung stellen wir das Browser-Game Selfaware Monopoly vor. Es besitzt die grundlegenden Funktionalitäten von dem bekannten Monopoly. Hinzu kommt jedoch noch eine weitere Komponente, die Regelbrüche anbietet. Wir erörtern die Frage inwiefern sich diese Komponente programmieren lässt, sodass alle Beteiligten einen Mehrwert erfahren. Der Programmcode ist frei verfügbar, inklusive Demonstrationsvideos. [1]

1 Einleitung

Informationssysteme ersetzen immer mehr elementare Bestandteile unserer Gesellschaft: Soziale Netzwerke, das Finanz- und Bankwesen, Wahlsysteme um nur eine wenige Bereiche zu nennen. Dabei vertrauen alle Beteiligten darauf, dass diese Systeme sich an *die Regeln* halten und alle Beteiligeten gleichermaßen anhand dieser Regeln behandeln.

Eine weiteres allbekanntes regealbasiertes System sind Gesellschaftsspiele. Ein populäres Gesellschaftsspiel, welches gleichzeitig auch eine vereinfachte Form unserer Gesellschaft abbildet, ist Monopoly.

Dieses Spiel haben wir uns zum Vorbild genommen, um ein System zu konzipieren, welches im Gegensatz zum regelkonformen Monopoly, Hintertüren anbietet um einzelne Regeln zu umgehen. Dafür haben wir Monopoly als Browserspiel implementiert mit einer zusätzlichen Komponente, die diese Hintertüren in Form von ***Deals*** öffnet.

Im Abschnitt 2 wird ein Konzept für die erweiterte Komponente des Monopoly's vorgestellt. Dabei wird definiert, welche Ressourcen und Informationen von Interesse sind und welche Plattformen als Ziel in Frage kommen. Die angebotenen Deals werden konzipiert und kategorisiert.

Im Abschnitt 3 wird die technische Implementierung des Prototypen beschrieben. Dabei werden wichtige Design Entscheidungen analysiert und der finale Stand des Prototypen dargestellt.

Im Abschnitt 4 wird versucht die in den vorherigen Kapiteln konzipierten und implementierten Methodiken in der Gesellschaft wiederzufinden. Dabei wird zum einen auf Softwatesysteme verwiesen, welche diese Methodiken nutzen, als auch auf gesellschaftliche Phänomäne, die der selben Struktur folgen.

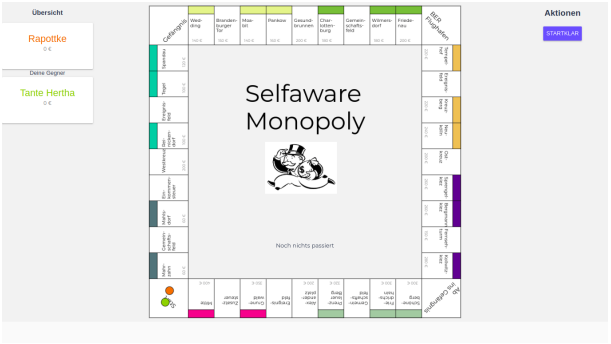
Im Abschnitt 7 wird beschrieben, welche Möglichkeiten und weitere Arbeiten aus der hier beschriebenen Arbeit entstehen könnten.

[1] https://github.com/ratnim/SelfawareMonopoly

HPI HASSO-PLATTNER-INSTITUT
Fachgebiet Computergrafische Systeme

Advanced Games of Life
Sommersemester 2018
Themenstellung und Anleitung: Willy Scheibel, Stefan Buschmann und Prof. Dr. Jürgen Döllner
http://www.hpi3d.de

Figure 1: *Beginn eines Monopolyspiels*



2 Konzept

Als das Spiel Monopoly in seiner aursprünglichen Form 1904 das erste mal aufkam, war es als sozial-kritisches Spiel gedacht. [2] Aufbauend auf dieser Grundidee wollen wir nun ein Spiel entwerfen, welches allen Spielern die Möglichkeit gibt die Spielregeln zu umgehen und sich im Gegenzug Information und Resourcen von den Spielern beschaft. Diese Komponente wird im folgenden ***Watson*** genannt.

2.1 Die Watson-Komponente

Watson bietet dem Spieler immer wieder die Möglichkeit, die Regeln von Monopoly zu umgehen und dem Spieler so einen Vorteil zu verschaffen. Im Gegenzug wird Watson versuchen unterschiedliche Ressourcen die nicht Inhalt des Spiels sind vom Spieler zu erhalten. Ein solcher Handel wird im folgenden als ***Deal*** bezeichnet. Die Spieler müssen aktiv die Entscheidung treffen, dass sie Watson nutzen. Dadurch kann Watson nur so weit agieren, wie es die Spieler zulassen. Sollten alle Spieler eines Spiels zu keiner Zeit die Deals von Watson annehmen, entspricht das Spiel einer regelkonformen Monopolypartie.

Der Name Watson ist frei gewählt und wurde von uns bewusst als Personifikation eines Algorithmuses gewählt. Assoziationen zu IBM-Watson, dem Supercomputer, der souverän Quizfragen beantwortet und Sherlock Homes Assistenten Watson sind vielleicht keine Zufälle.

2.2 Watsons' Ziel

Watson interessieren persöhnliche Informationen, Ressourcen und die Nutzung der Identität der Spieler, um in ihrem Namen aufzutreten.

Persöhnliche Informationen, die Watson bekommen möchte, sind zum Beispiel die Identität des Spielers auf Sozialen Platformen oder Zugriff auf andere Dienste wie E-Mail Services.

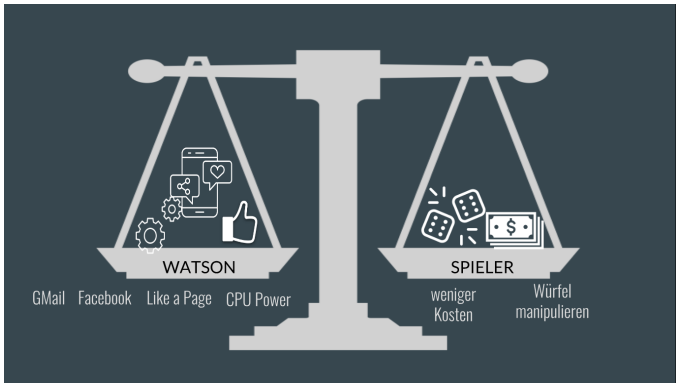
[2] https://winningmoves.de/blog/die-geschichte-des-monopoly-spiels, Stand: 01.01.2019

Ressourcen die Watson bekommen möchte ist Rechnenleistung um Bitcoinmining zu betreiben.

Auf welche Dienste dabei zurückgegriffen wird, wird in 2.6 erörtert.

2.3 Angebotene Deals

Figure 2: *Konzept visualisiert: Abwägen zwischen Mehrwert im Spiel und Preigeben von Informationen und Ressourcen.*



Watson bietet dem Spieler unterscheidliche Deals an, um die Regeln zu umgehen. Dabei gibt es verschiedene Arten von Deals.

	verdeckt	**offensichtlich**
direkt	Würfel manipulieren	-
indirekt	Glück beeinflussen	Baupreis reduzieren

Table 1: *Die Tabelle kategorisiert die Effekte der Deals, die durch Watson angeboten werden, in zwei Dimensionen. In der horizontalen, ob der Effekt eines Deals für die Spieler sichtbar ist und in der vertikalen, ob der Effekt direkt eintritt.*

Tabelle 1 kategorisiert die Effekte eines Deals in zwei Dimensionen. *Direkt* oder *indirekt*, das bedeutet ob der Spieler einen direkten Nutzen bekommt und in *verdeckt* und *offensichtlich*, was bedeutet, ob die anderen Mitspieler mitbekommen, ob soeben ein Deal gegriffen hat. Daraus ergeben sich strukturell folgende Deals.

Offensichtlich-direkt wäre ein Deal, der für alle Mitspieler sichtbar von statten geht, wie z. B. der Spieler bekommt Geld aus der Bank. Da es in dem Szenario nicht darum geht, wer am skrupelosesten schummelt, wird diese Art von Deals nicht durch Watson angeboten.

Offensichtlich-indirekt sind alle Deals, bei denen andere Spieler den Effekte als Regelbruch ausmachen können, aber die nicht direkt beim Preisgeben von Informationen oder Ressourcen einen Effekt haben. Der Spieler weiß zu den Zeitpunkt, wenn ihm der Deal angeboten wird nicht, ob er die Leistung durch Watson braucht. Dadurch soll der Spieler zum *Hammstern* angehalten werden und im Zweifel mehr von sich preis zu geben als nötig wäre.

Watson bietet den Spielern zum Beispiel an, Bitcoin-Mining zu betreiben und so für jede Spielrunde, die das Bitcoin-Mining im Hintergrund läuft, einen so genannten ***Watson-Coin*** zu generieren. Die Watson-Coins werden dann durch Watson für jeden Spieler in einem Konto gespeichert. Der konkrete Stand dieses Kontos ist für den Spieler aber nicht einsehbar. Baut ein Spieler, der Watson-Coins besitzt, Häuser, so wird der Preis pro Watson-Coin um 50 Euro verringert.

Für andere Spieler ist es nur beim Hausbau einsehbar, ob der Spieler einen Deal von Watson angenommen hat oder nicht. Die Zahlen werden innerhalb des Spiels aber nur durch genaues hinschauen und nachrechnen einsehbar sein, wodurch es erschwert ist dies von außen mitzubekommen.

verdeckt-direkt sind alle Deals, bei denen der Spieler Informationen oder Ressourcen an Watson übergibt und einen sofortigen Nutzen davon hat. Durch den direkten Mehrwert, den der Spieler gezeigt bekommt, soll der Spieler in Versuchung gebracht werden, selbst wenn er eigentlich keine Deals annehmen möchte, dies an dieser Stelle doch zu tun. Zum Beispiel ist es gerade im frühen und im späten Spielverlauf von entscheidender Bedeutung, entweder unbedingt oder auf keinen Fall ein bestimmtes Würfelergebniss zu erzielen.

Watson bietet deshalb den Spielern an die Würfel zu manipulieren. Der Spieler muss dafür persönliche Daten an Watson übergeben und darf sich so sein nächstes Würfelergebniss wählen. Für andere Spieler ist es in keiner Weise ersichtig, ob der Spieler einen Deal von Watson angenommen hat oder nicht.

verdeckt-indirekt sind alle Deals, bei denen der Spieler Informationen oder Ressourcen an Watson übergibt aber keinen direkten Effekt hat. Der Spieler weiß dabei nicht einmal, ob es überhaupt einen Effekt hat. Da der Effekt nicht einmal für den Spieler selbst auszumachen ist, haben andere Spieler keine Möglichkeit herrauszufinden ob der Spieler Deals von Watson annimmt oder nicht.

Watson bietet den Spielern zum Beispiel an bestimmte Webseiten zu besuchen oder Kommentare und Seiten auf Facebook zu liken. Tut der Spieler dies, merkt sich Watson dies und beeinflusst die Ereignisskarten, die ein Spieler bekommt, wenn er auf ein Ereignissfeld läuft.

2.4 Timing des Eingreifens

Es ist wichtig das Watson für andere Spieler nicht merklich in den Spielfluss eingreift. Das heißt, wenn ein Spieler mit Watson interagiert sollen es die anderen Spieler nicht mitbekommen. Dafür sollte Watson möglichst dann die Deals anbieten, wenn der Spieler gerade im Spiel nichts zu tun hat. Deshalb bietet Watson Deals eher Spielern an, die gerade nicht an der Reihe sind.

Desweiteren soll Watson eine hohe Erfolgsrate haben, das heißt, die Wahrscheinlichkeit, das ein Spieler den Deal annimmt soll hoch sein. Um dies zu erreichen, muss auf unterschiedlichste Faktoren geachtet werden.

Es wird bestimmt, ob der Spieler den Effekt des Deals gebrauchen kann oder nicht. Beispiele: Einem Spieler der sehr viele Straßen besitzt, anzubieten die Würfel zu manipulieren hat wahrscheinlich weniger Erfolgschancen, als einem Spieler der unbedingt diese eine Straße braucht. Einem Spieler mit einem unbebauten Straßenzug den Baupreis zu reduzieren, hat wahrscheinlich eine höhere Chance als einem Spieler ohne Straßen.

Spieler die schon Deals angenommen haben sollen verhältnissmäßig viele Deals angeboten werden. Wohingegen Spieler die noch keine Deals angenommen haben weniger bis garkeine Angebote mehr bekommen.

Deals sollen sich durch nur temporäre Verfügbarkeit interessant machen. Somit ist der Spieler angeregt, selbst wenn er den Effekt erst später nutzen kann, diesen auch anzunehmen, weil er im Zweifel nicht darauf verzichten will.

Außerdem sollen einige Deals zufällig angeboten werden. Das heißt ein Spieler bekommt eigentlich einen Deal auf denen eine

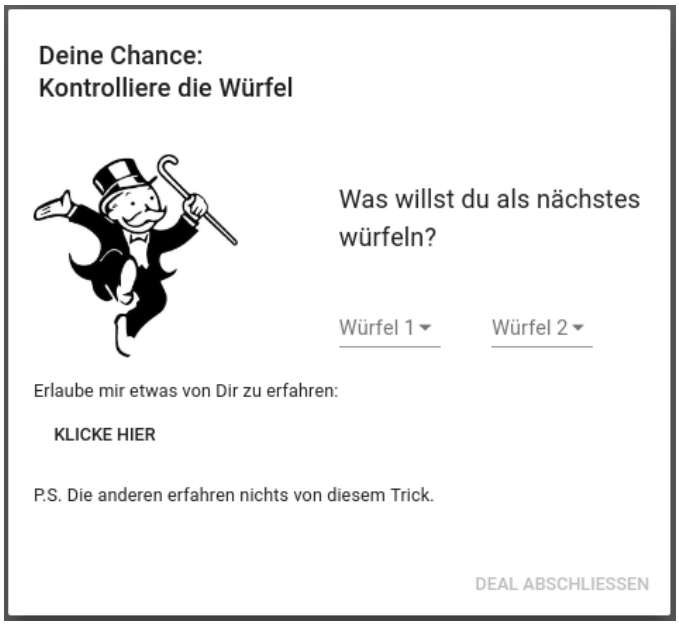
oder mehrere der definierten Kriterien nicht zustimmen.

Deals werden regelbasiert angeboten. Die Regeln, die wir aufgestellt haben sind:

1. ***isNotMyTurn*** heißt, dass Deals nur angeboten werden, wenn der Spieler nicht am Zug ist. Der Grund für diese Regel ist, dass es zu keiner Zeitverzögerung des Spielzuges kommen soll, wodurch andere Spieler einen Dealabschluss vermuten könnten. Außerdem bleiben so Spieler selbst wenn sie nicht am Zug sind in das Spielgeschehen involviert.
2. ***"roundsSinceLastInteraction > 3"*** heißt, dass mindestens drei Runden gespielt werden müssen bevor erneut ein Deal angeboten wird. Die Drei ist ein von uns gewählter Threshold. Binnen drei Runden schafft ein Spieler durchschnittlich etwa eine Halbe Umrundung des Spielfeldes.
3. ***"Probability(ranking, behaviour, random) > 0.4"*** heißt, das eine Wahrscheinlichkeit berechnet wird basierend auf den drei Parametern ranking, behaviour und random. ranking ist die aktuelle Position des Spielers im vergleich zu den anderen. Dies berechnet sich aus Kontostand und Wert der Straßen. behaviour ist ein Faktor, der das Spielverhalten beschreibt. Wurden bisher viele Deals angenommen ist der Faktor sehr hoch. random ist ein Faktor, der die Funktion nicht deterministisch macht. Wird der von unse gegeben Threshold von 0.4 überschritten, so wird ein Deal vorgeschlagen.

2.5 Watson-Spieler Interaktion

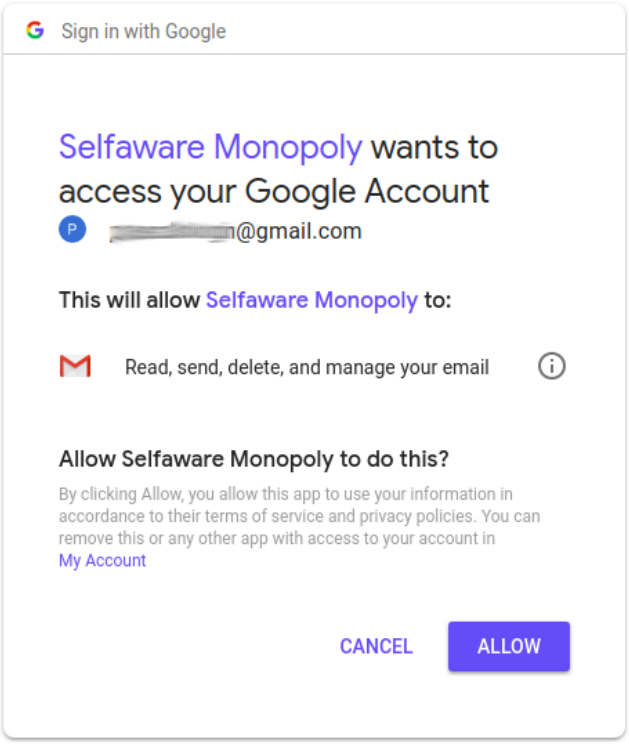
Figure 3: *Watson Dialog: Beeinflussen der Würfel*



Die Watson Komponente soll zwar Teil des Spiels sein, aber für den Spieler soll Watson eine klar erkennbare Komponente innerhalb des Interfaces darstellen. Dadurch soll klar werden, dass dies nicht mehr zu dem eigentlichen Monopolyspiel gehört und der Spieler an dieser Stelle eine Grenze überschreitet. Der Spieler muss dadurch aktiv entscheiden, ob er diesen Teil des Spiels nutzt.

Watson erscheint als ein Dialogfenster. Dabei gibt es in der Eingabemaske dem Spieler zunächst den Vorteil bekannt, sowie mögliche Konfigurationsparameter, wie z. B. Würfelergebnis.

Figure 4: *Weiterleitung zum OAuth Dialog des API Anbieters. In diesem Fall werden Berechtigungen von GMail eingefordert.*



Darunter folgt die Aufforderung eine Aktion auszuführen. Die Aktion ist in der Regel mit einem Klick ausführbar.

2.6 Arten von Deals

Für die Aktionen haben wir auf Dienste von Drittanbietern konzentriert, die APIs bereitstellen, welche persönliche Informationen oder Aktionen zulassen. Folgende Dienste haben wir in Betracht gezogen: Amazon, Twitter, Facebook, Instagram, LinkedIn, Spotify, Google Mail und PayPal. Eine Analyse der API Dpkumentationen hat folgendes ergeben. Amazon hat keine für uns nützliche API. LinkedIn bietet lediglich den Usernamen. Bei allen anderen Diensten ist es möglich im Namen des Spielers, sofern er die Berechtigung freigegeben hat, zu handeln. Um Zugriffsrechte zu bekommen wird die von allen Diensten angebotene OAuth2.0 Schnittstelle verwendet. OAuth ist ein Prozess bei dem sich ein Nutzer bei einem Dienst anmeldet und anschließend ein Zugangsschlüssel (sog. *access token*) bereitgestellt wird, welcher mit ausgewählten Berechtigungen versehen ist. Mit diesem Zugangsschlüssel kann die authorisierte Anwendung dann Daten abfragen oder Aktionen ausführen.

Im Abschnitt 4 gehen wir genauer auf die Informationen und Aktionen die abgegriffen bzw. ausgeführt werden können ein und wie Nutzer auch dem entgegen wirken können.

Ein weiterer Nutzen neben den Informationen sind Ressourcen. Hierfür haben wir uns vor allem auf das Thema Bitcoinmining konzentriert. Beim Bitcoinmining wird die Kryptowhärung Monero gemined. Dafür wird die Coinhive API verwendet, welche Bitcoint-Mining im Browser ermöglicht. [3] Das tatsächlich er-

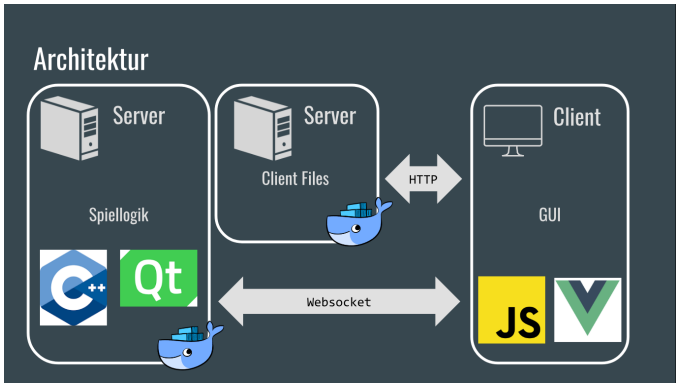
[3] https://coinhive.com/

wirschaftete Guthaben geht an Watson, bzw. an das System, bzw. an die Autoren dieses Spiels.

3 Vorstellung Prototyp

Der Prototyp präsentiert sich dem Nutzer als Website, welche dieser in einem normales Webbrowser aufrufen kann. Die Bereitstellung als Website bietet die Vorteile, dass einerseits jedweder Nutzer das Produkt nutzen kann, ohne dass spezielle Voraussetzungen, wie beispielsweise ein spezifisches Betriebssystem, gegeben sein müssen, während andererseits der Server als Anbieter unter bekannten und kontrollierbaren Bedingungen entwickelt werden kann.

Figure 5: *Softwarearchitektur visualisiert*



Der Server selbst ist geteilt in einen HTML-Server, welcher die Websites bereitstellt, sowie einen Websocket-Server, welcher die Spiellogik, inklusive Manipulationen implementiert und Auskunft über Ereignisse in den laufenden Spielen gibt. Diese beiden Komponenten sind zum großen Teil unabhängig. Die einzigen Verbindungen bestehen in der API, mit der über die Websockets kommuniziert wird, sowie dem Umstand, dass der Client die korrekte Adresse für die Verbindung zum Websocket kennen muss. Entsprechend dieser klaren Aufteilung wurden die Implementierungen parallel von zwei getrennten Gruppen durchgeführt, aufbauend auf der gemeinsam genutzten Websocket API.

Da im finalen System mehrere Akteure zusammenwirken, kommt das Problem der Synchronisation auf. In diesem speziellen Fall müssen sich der Server und sämtliche an einem Spiel beteiligten Clients einig sein über den Zustand dieses Spiels. Für jede Änderung einmal den vollständigen Zustand an sämtliche Clients zu senden wäre nicht nur eine zusätzliche Last für das Netzwerk, sondern auch ein erheblicher Mehraufwand für den Server, der zusätzlich dadurch erhöht wird, dass bedingt durch die Manipulationen von Watson manche Informationen nur bestimmten Spielern zugänglich sein dürfen. Aus diesem Grund müssen sämtliche Teilnehmer eine eigene lokale Kopie des Zustandes vorhalten, welche sie bei Bedarf aktualisieren. Der individuelle lokale Zustand wird ausgehend von einem Grundzustand iterativ durch Anwendung einzelner lokaler Aktionen erzeugt. Während der Grundzustand einzig von der Anzahl der beteiligten Spieler, deren Namen und dem gewählten Spielbrett abhängt, behandeln die Aktionen sämtliche Änderungen, welche durch die Züge und Entscheidungen der Spieler entstehen. Um inkrementellen Fehlern vorzubeugen, wird zum einen darauf geachtet, dass Aktionen in strikter Reihenfolge angewendet werden, in der sie aufgetreten sind. Zum anderen wird für jede Änderung eines Wertes nicht die relative Veränderung übermittelt, wie es vor allem für die Kontostände und die Positionen der Spielfiguren einleuchtend wäre, sondern das absolute Ergebnis der Aktion.

Die Abfolge der Aktionen wird für jedes Spiel in einer Datenbank hinterlegt. Ursprünglich erfolgte dies nur zur Dokumentation der mit dem Prototypen durchgeführten Nutzerstudien. Später wurden diese Aufzeichnungen auch dafür verwendet, um es Spielern, die durch äußere Störungen die Verbindung zum Server verloren haben, zu ermöglichen, wieder in ihr Spiel einzusteigen und die „verpassten“ Aktionen nachzuholen.

Die Kommunikation über die Websockets erfolgt asynchron, da beide Kommunikationspartner Meldungen an den anderen machen können, welche meist keiner direkten Rückmeldung bedürfen. Als Datenformat wurde JSON gewählt, da es einerseits von beiden Seiten einfach erzeugt ist und leicht dekodiert werden kann, und andererseits selbst im „kodierten“ Zustand noch menschenlesbar und somit leichter nachvollziehbar ist. Der Inhalt einer Nachricht besteht aus einem Feld für den Zweck sowie weiteren Informationen, sofern der diese für den Zweck relevant sind. Während beispielsweise die Meldung über das Ende des Spiels ohne weitere Parameter versendet werden kann, wäre eine Nachricht über die Bewegung eines Spielers ohne die Angabe des Ziels sinnlos.

Die Inhalte der Nachrichten reichen von organisatorischen Belangen, wie der Anmeldung eines Nutzers oder dem Beitritt zu einem Spiel, über spielinterne Geschehnisse, wie die Augenzahl der Würfel nach einem Wurf oder der Veränderung des Kontostands eines Spielers, bis hin zur Ausführung von Schummelaktionen, wie der Festlegung des nächsten Würfergebnisses. Während beide Kommunikationspartner spontan Nachrichten an den anderen schicken können, übernimmt der Client, also der Nutzer mit seinem Browser, eine aktivere Rolle als der Server. Während der Server nur Statusmeldungen an die Clients schickt, wird ein Client bei bestimmten Anfragen auf spezifische Meldungen des Servers warten. Beispielsweise erwartet ein Client, der die Nachricht zum Kauf eines Grundstücks gesendet hat, dass einerseits der Besitzer dieses Grundstücks geändert wird, und sich andererseits der eigene Kontostand ändert. Zu beachten ist, dass jede dieser Anfragen genauso gut mit einer Fehlermeldung beantwortet werden kann. Diese Fehlermeldungen handeln dabei von nicht durch Watson abgedeckten Regelverstößen, wie fehlende Besitzansprüche oder einen zu geringen Kontostand.

Der HTML-Server stellt den Nutzer zwei verschiedene Webseiten bereit. Die erste Seite, die ein Nutzer sieht, ist die Lobby, in der er sich erst einen Anzeigenamen aussuchen darf, und anschließend entweder einem bestehenden Spiel beitreten oder ein neues Spiel einleiten kann. Die andere Seite behandelt eine konkrete Instanz des Spiels. Sie enthält Informationen über den Zustand des Spielbretts, die Besitztümer aller beteiligten Spieler, sowie eine Auflistung sämtlicher Aktionen, die der Nutzer ausführen kann. Es liegt in der Natur dieser beiden Seiten, vor allem aber in der zweiten Seite, dass sie dynamischen Inhalt zeigen müssen, der sich während der Betrachtung durch äußere Einflüsse ändern kann. Diesen dynamischen Inhalt auf Serverseite einzusetzen und die fertige Seite an den Client zu senden, hätte nicht nur dazu geführt, dass sich die Seiten aufs Geratewohl erneut laden müssten, sondern auch den HTML-Server als Mittelsmann zwischen dem Client und dem für die Spiellogik verantwortlichen Server eingesetzt. Stattdessen werden beide Seiten mit einem statischen Grundgerüst ausgeliefert, welches mittels Skripten zuerst eine direkte Verbindung zum Websocket-Server aufbaut, um dann mit den von dort empfangenen Daten die eigentliche Seite aufzubauen. Die Verwendung von Skripten und Websockets erlaubt nicht nur, die Aktualisierung der angezeigten Inhalte zu verzögern, bis auf dem Websocket tatsächlich neue Informationen eintreffen, sondern zusätzlich punktuelle Änderungen der betroffenen Abschnitte, ohne die Seite vollständig neu laden zu müssen.

Für die dynamische Anpassung der Seiten kommt das Framework

Vue[4] zum Einsatz. Es erlaubt, die Struktur des HTML-Dokuments und die Eigenschaften seiner Elemente an Skript-Variablen zu binden, sodass sich der Inhalt nach dem Wert der Variablen richtet. Zusätzlich werden Callbacks eingerichtet, mit denen sämtliche von einer Variablen abhängigen Elemente automatisch aktualisiert werden, sobald sich der Wert der Variablen ändert. Dies geschieht separat für alle gebundenen Variablen, sodass der Aufwand für die Aktualisierung gering gehalten wird. Zusätzlich werden vielerorts die mit HTML 5 eingeführten Templates[5] verwendet. HTML-Templates erlauben es, HTML-Strukturen zu definieren, welche kopiert und angepasst werden können, bevor sie in das sichtbare Dokument eingefügt werden. Durch das Zusammenwirken von Templates und den Manipulationen durch Vue erhalten die Seiten einen ausgeprägten objektorientierten Charakter.

Die Architektur des Websocket-Servers basiert auf dem Modell des erweiterten endlichen Automaten. Jedes laufende Spiel wird dabei durch einen Automaten modelliert, welcher auf Basis der von den Clients empfangenen Eingaben Zustandsänderungen vollführt und Ausgaben erzeugt. Während das Spielfeld und die Besitztümer der Spieler in jederzeit erreichbaren Variablen gehalten werden, werden die Zustände des Automaten überwiegend durch die verschiedenen Phasen während des Zuges eines Spielers bestimmt. Diese Zustände bestimmen die Auswahl an Aktionen, die die Spieler zum entsprechenden Zeitpunkt ausführen können, und basieren dabei auf den Regeln des Spiels. Beispielsweise darf ausschließlich der Spieler, der gerade am Zug ist würfeln, und dies auch nur, wenn er seinen Zug gerade begonnen hat oder durch einen Pasch erneut würfeln darf. Zustandsübergänge erfolgen abhängig von den Aktionen der Spieler und dem aktuellen Spielstand, und folgen ebenfalls den Spielregeln. Die Modellierung mittels Zuständen erlaubt eine übersichtliche Implementierung einzelner Aktionen, und schützt gleichzeitig davor, ungültige Aktionen zu unternehmen.

Während die Spiellogik mit dem genannten Automatenmodell von Grund auf implementiert wurde, wurden für die Kommunikation mit den Clients entsprechende Bibliotheken benutzt. Speziell finden einige Komponenten des Frameworks Qt[6] Anwendung. Qt bietet neben einer Implementierung von Websockets auch die Möglichkeit zur asynchronen Kommunikation mittels Signalen und einer Event Queue, wovon in Bezug auf das Versenden der Rückmeldungen an die Clients Gebrauch gemacht wurde. Für die Datenbank, in der sowohl die Spieler als auch sämtliche Spiele inklusive ihrer Abläufe hinterlegt werden, bietet Qt eine Auswahl von Komponenten, welche die Verwendung einer SQL-Datenbank erleichtern.

Die Schummelkomponente „Watson“ befindet sich hauptsächlich im Client. Während die Effekte, die Watson herbeiführt, im Server umgesetzt werden müssen, werden die Entscheidungen, ob und welche Angebote Watson dem Nutzer macht, lokal im Client getroffen. Auf der anderen Seite der Angebote, speziell bei der Anmeldung bei Dritten, besteht das Problem, dass nutzerspezifische Daten vom Client zum Server übertragen werden müssen. Die Idee, die Zugangsdaten an den Server zu senden und über die API des Drittanbieters an entsprechende Zugangstokens zu gelangen, wurde davon abgelöst, den Client die Tokens direkt anfragen zu lassen, um diese dann an den Server zu senden. Der letzte Schritt, das Senden an den Server, wird aus Sicherheitsgründen unterlassen. Für die Untersuchung des Verhaltens der Nutzer reicht der Erhalt der Tokens vom Drittanbieter aus.

Der Prototyp in seinem gegenwärtigen Zustand bildet nicht das vollständige Regelwerk eines handelsüblichen Monopoly ab, da einige teils elementare Bestandteile, wie Auktionen oder der Handel zwischen Spielern, fehlen. Trotzdem ist er lauffähig, und dazu in der Lage, vollständige Spiele von Beginn bis Ende durchzuführen. Aufgrund unvorhergesehener Verzögerungen war es nicht möglich, Nutzerstudien mit dem Prototypen durchzuführen.

4 Diskussion

Wie bereits in 2.6 angesprochen, macht sich die Watson Komponente die APIs verschiedener Dienste zunutze. In diesem Abschnitt wird diskutiert, welche Rechte über die APIs freigegeben werden.

Durch den Prototypen wurde gezeigt, das es möglich ist Systeme zu bauen, die versuchen den Nutzer mit seiner Zustimmung dazu zu bringen Information und Ressourcen von sich herzugeben ohne einen Mehrwert außerhalb des bereitgestellten Spiels zu gewähren.

4.1 Bitcoin-Mining

Eine der Gegenleistungen, die für Schummelaktionen verlangt werden, ist die Verwendung von Rechnerressourcen des Nutzers, speziell für das sogenannte „Mining“ von Kryptowährung. Um das Mining durchzuführen, werden Skripte von Coinhive[7] in Anspruch genommen, welche die Währung Monero (XMR)[8] erzeugen. Monero ist dem weitaus bekannteren Bitcoin sehr ähnlich, und steht zum Zeitpunkt des Verfassens dieser Ausarbeitung mit einem Wechselkurs von rund 113 USD zu 1 XMR im Kurs.

Das Mining für Monero erfolgt mittels einer Blockchain. Transaktionen, also Übertragungen von Monero von einem Besitzer zu einem Anderen, werden von den Minern in Blöcken verifiziert, und diese Blöcke der Blockchain angehängt. Um übermäßiger Inflation vorzubeugen, wird die Erstellung gezielt künstlich erschwert, auf dass durchschnittlich nur ein einziger Block alle 2 Minuten entstehen kann. Die künstliche Hürde besteht darin, einen Block zu generieren, dessen Hashwert unterhalb eines gewissen Schwellwerts liegt. Jedem erfolgreich angefügten Block liegt ein stetig schrumpfender Bonus bei, der dem Erzeuger des Blocks gutgeschrieben wird.

Aufgrund der Seltenheit der Blöcke und der damit verbundenen Boni, ist das Konzept der „Pools“ entstanden. Pools sind Zusammenschlüsse mehrerer Interessenten, die ihre Ressourcen zum Mining benutzen. Erhält einer der Beteiligten den Bonus, wird er unter den Teilnehmern des Pool aufgeteilt.

Coinhive bietet einen derartigen Pool an, und nutzt mittels der bereitgestellten Skripte die Rechner Dritter für das Mining. Die Boni aus erfolgreichen Blöcken behält Coinhive ein, zahlt aber demjenigen, der die Skripte auf seiner Seite einsetzt, einen kleinen Betrag pro Hashwert, den die von ihm ausgelieferten Skripte ausprobieren. Die Höhe dieses Betrags hängt ab von der Wahrscheinlichkeit, einen gültigen Block zu erzeugen, und von der Höhe des Bonus, den der Block einbringen würde.

Laut Angaben von Coinhive zum Zeitpunkt der Schriftsetzung betrugen die Erfolgswahrscheinlichkeit rund 1 zu $7,155 \cdot 10^{12}$ und der Bonus pro Block rund 3,8 XMR. Zudem geben sie an, ihre Skripte ließen auf hochwertigen Prozessoren rund 90 Hashwerte pro Sekunde berechnen. Nimmt man all diese Zahlen, inklusive des Wechselkurses, zusammen, erhält man, dass ein einzelner Nutzer innerhalb einer Stunde einen Gegenwert von rund 0,000013 USD erzeugt.

[4] https://vuejs.org
[5] https://www.w3.org/TR/html50/scripting-1.html#the-template-element
[6] https://qt.io
[7] https://coinhive.com
[8] https://getmonero.org/

4.2 Facebook API

Facebook ist mit über 2000 Millionen Nutzern [9] die größte soziale online Platform. Sie ermöglicht den Austausch mit Freunden und Fremden. Als Entwickler kann man sog. Apps für Facebook schreiben. Folgende Informationen können dabei erfragt werden : E-Mail, Name, Alter, Geschlecht, Geburtstdatum, Likes und weitere Außerdem ist es möglich einer Seiten einen sog. *Like* zu geben. Folgende Informationen und Aktionen sind u.a. nicht möglich: Freundesliste abfragen und persönliche Nachricht schreiben.

Die sammelbaren Daten reichen aus, um ein sehr genaues Profil einer Person zu erstellen. Abhängig davon wie aktiv und seriös eine Person auf facebook unterwegs ist, können neben den persönlichen Metadaten wie Name, Wohnort etc. auch auf politische Interessen, Musikgeschmack, Freizeitaktivitäten und Lieblingsvereine geschlossen werden. Mit diesen Informationen kann einem Nuzer noch genauere Werbung zugespielt werden, aber auch für statistische Interessen sind die Informationen wertvoll.

4.3 GMail API

GMail ist der weit verbreitete Mail-Service von Google. Mit der GMail API bekommt man Zugriff auf Lese, Schreibe und Verwaltungsrechte des E-Mail-Postfaches. Diese Berechtigungen werden u.a. von Mail-Clients wie Thunderbird oder Outlook benötigt. Wenn Anwendungen mit anderen Interesse Zugriff auf das E-Mailkonto bekommen, so können ebenfalls viele Details über eine Person bezogen werden. Eine Nutzung sind Bestellbestätigungen, die Aufschlus auf Kaufverhalten und Vermögenssituation zulassen.

5 Analogie zu unserer Gesellschaft

Innerhalb des Prototypens werden durch Watson immer wieder Möglichkeiten angeboten die allgemein geltenden Regeln zu umgehen. In unserer Gesellschaft gelten genauso wie in dem Monopoly Spiel Regeln die für alle gültig sind.

Die durch Watson angeboten Deals wurden in 4 Kategorien eingeteil, wobei für den Prototyp nur drei von relevanz sind. Alle diese Kategorien von Deals lassen sich auch in der wirklichen Welt finden.

5.1 Verdeckte-Direkte Deals in der Gesellschaft

Verdeckte-Direkte Deals sind Deals, bei dennen der Nutzten zwar verdeckt bleibt und es nicht klar ist, ob wirklich geschummelt wurde. Innerhalb des Prototypens ist dies das Manipulieren des Würfelergebnisse.

Analog dazu findet man in unserer Gesellschaft Schmiergeld, Bestechungen und Korruption. Dabei werden Personen, Firmen oder Instutionen, welche in der Position sind Entscheidungen im Interesse aller zu treffen durch Geld oder Gegenleistungen beeinflusst. Dies kann bei Bau- oder Eventvergaben, bei politischen Entscheidungen oder aber auch das wegschauen bei einer ilegalen Handlung sein. Im nachhinein ist es dann nur möglich anhand der Gegenleistung festzustellen ob es wirklich Bestechung bzw. Korruption war oder nicht.

Bestechung und Korruption sind überall in unserer Gesellschaft zu finden. Sei es im Entertainment Sektor wie bei der Vergabe der Austragungsorte für die WM Frankreich und Südafrika wo es zu Korrupierung gekommen ist. [FAZ 2015]

[9] laut Facebook Report

Innerhalb der Politik und selbst im der Exekutive ist Korruption ein Thema. Zum Beispiel wurde jüngst Interpol-Chef Meng-Hongwei vor Bestechungsgelder angenommen zu haben. [NTV 2018]

So gennante PayToWin-Games sind bekannte vertreter von Informationssystemen, in welchen verdeckten-direkte Deals genutzt werden. Bekannte Beispiele dafür sind Tribal Wars 2 oder Grepolis. Die Spiele bieten dem Spieler die Möglichkeit, erweiterte Funktionalität und Inhalte des Spiels gegen Echtgeld zu erhalten. Dabei kann der Spieler zum Beispiel nur eine beschränke Anzahl an Gebäuden in Auftrag geben. Durch bezahlung Echtgeld kann er diese Beschränkung aufgehoben. Dadurch hat er den Vorteil mit weniger Aufwand zu bauen. Für andere Spieler ist es nicht einsichtig ob er dieses Feature in Anspruch nimmt oder nicht.

5.2 Offene-Indirekte Deals in der Gesellschaft

Offene-Indirekte Deals sind Deals, bei welchen der Nutzen für den Spieler verzögert eintritt. Außerdem ist für andere Spieler erkenntlich, dass die Regeln umgangen wurden. Im entwickelten Prototypen bietet Watson den Spielern an durch Bitcoin-Mining den Preis beim Bau von Häusern zu reduzieren.

In der Geselschaft lassen sich diese Effekte kaum von den verdeckten indirekten Deals unterscheiden. Dabei werden oft Menschen bewertet um später dann anhand dieser Bewertung Privilegien bekommen oder nicht.

Ein Beispiel in unserer Gesellschaft wäre dabei die Schufa Auskunft. Diese Auskunft wird an vielen Stellen genutzt um dann eine Entscheidung über einen Mietvertrag oder einen Kreditvertag zu vergeben.

Abstractism ist ein Spiel welches auf Steam veröfftlich wurde und wie im entwickelten Prototyp Bitcoin-Mining nutzt. [Abs 2018] Dabei hat der Spieler aber nichts von dem Bitcoin Mining erfahren und auch keine Gegenleistung erhalten.

5.3 Verdeckte-Indirekte Deals in der Gesellschaft

Verdeckte-Indirekte Deals sind Deals, bei denen der Nutzen verzögert eintritt und auch nicht erkenntlich ist ob die Regeln umgangen wurden oder nicht. Im entwickelten Prototypen ist es möglich durch besuchen von Websites die Ereignisskarten zu beeinflussen.

In der Gesellschaft lassen sich solche Phänomäne immer wieder erahnen aber nur selten feststellen. Die Handlung ist oft sehr menschlich und folgt dem Prinzip die eine Hand wäscht die andere. Hat mir jemand etwas gutes getan dann helfe ich ihm in der Regel auch. Es könnte natürlich sein das ich ihm auch so helfe. Problematisch wird es erst, wenn dies zu einer klaren Verzerrung wird, bzw. zur dauerhaften Bevorzugung einzelner führt.

Ein-Parteien-Staaten folgen diesem Schema. Alle Mitglieder der Partei haben meist drastische Vorteile obwohl sie theoretisch genau die gleiche Banhandlung wie alle anderen erhalten. Ein prominetes Beispiel dafür war die DDR. Ein moderne und digitalisierte Form davon ist in China zu finden. Dort sollen mit einem Überwachungssystem bis zum Jahr 2020 1.35 Millionen Menschen überwachen und bewerten werden. [Chi 2018] Die so bewerteten Bürgen, bekommen dann anhand ihrer Handlungen ein Rating, welches sie dann in ihren Möglichenkeiten beschränkt oder priveligiert.

Da eine solche Art von Deals einer sehr menschlich Angewohnheit folgen, ist es schwer zu sagen wie weit die Politik davon beeinflusst ist. Aber es ist immer wieder zu sehen das Politiker für eine

Entscheidung stimmen die sie eigentlich nicht vertreten, weil andere Politiker denen sie gefällig sind dafür stimmen.

6 Warum baut man solche Systeme?

Wie in vorangegangen Teil des Kapitels 4 gezeigt wurde lassen sich alle implementierten Methodiken sowohl in bestehenden Software Systemen als auch in der Gesellschaft wiederfinden.

Dabei implementieren dieses Systeme meist nicht alle der Methodiken die im dem hier vorgestellten Prototyp gezeigt wurden, sondern nutzen eine Teilmenge mit einer der folgenenden Motivationen.

6.1 Kapitalerzeugung

Viele Systeme sind konzipiert um direkt Kapital zu erzeugen. Manche davon nutzen die hier vorgestellten Methodiken um dies zu erreichen. Innerhalb der Computerspiele Industrie lasseb sich so gennante pay-to-win Spiele finden. Diese ermöglichen es dem Spieler mit Echtgeld die Regel zu umgehen oder einen direkte Vorteil innerhalb des Spiels zu erkaufen. Andere Spiel wie Abstraction aber auch Anwendungen wie Sweat-Coin nutzen Bitcoin-Mining um mit der Rechenleistung des Spielers Kapital zu generieren.

6.2 Informationsbeschaffung

Systeme welche das Ziel haben, Informationen über mögliche Kunden oder das Nutzungsverhalten von bestimmten Zielgruppen offen zu legen finden sich überall. Die Teilmenge der Systeme, welche dafür einen Bonus oder Vorteil dem Nutzer geben, welche sonst nicht oder nur gegen Echtgeld erhältlich wären entsprechen der Methodik des Prototypen. Beispiele dafür sind endlos, angefangen bei Gratiszeitungen gegen die eigene Anschrift bis hin zum verlinken des Facebook Accounts um einen Startbonus beim Anmelden auf einer Plattform zu bekommen.

6.3 Meinungsverzehrung

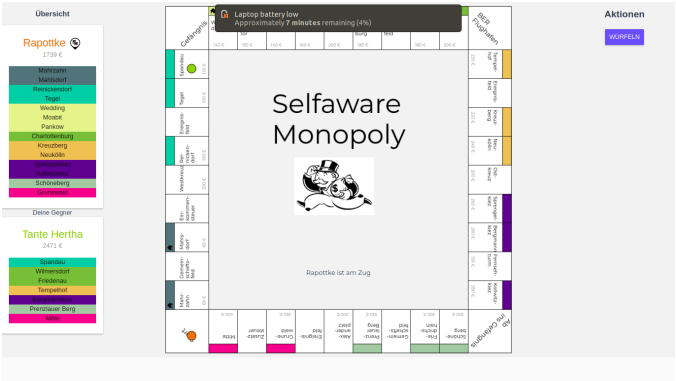
Beim Einkauf im Internet ist die Bewertung eines Produkts oft das wichtigste Kriterium ob man etwas kauft oder nicht. Umso breiter das Spektrum der Bewerter desto eher vertraut man der Bewertung. Änlich verhält es sich auch mit Bewertungen für Veranstaltung, Filme oder Restaurants. Angenommen eine Firma will mehr oder bessere Berwertung machen ohne dies zu forcieren, kann das ein endlosses Unterfangen werden. Deshalb wird versucht Nutzer zu einer Bewertung zu bewegen. Ab dem Zeitpunkt wo der Nutzer für eine positive Bewertung eine Gegenleistung bekommt oder eine Berwertung vorgegeben wird verlässt man den legalen Bereich.

Wird die Identität einer anderen Person genutzt um einen Like oder eine Bewertung zu hinterlassen, ist dies gänzlich ilegal. Facebook geht gegen so gennante FakeLikes aggressive vor. Durch Facebook wird die im Prototyp umgesetzte Methode als Klickfarm bezeichnet. [Fak 2018]

7 Future Work

Inhalt dieser Arbeit war es ein System zu erschaffen welches mit der aktiven Beteiligung der Nutzer bestimmte Regeln bricht und das System im Gegenzug versucht so viele Information und Ressourcen des Nutzers zu erhalten.

Figure 6: *Verlauf des Spiels inklusive Auswirkungen des Coin-Minings.*



7.1 Inteligente Watson Komponente

Die Entscheidung wann welche Deals durch die Watson Komponente angeboten werden erfolgt momentan noch regelbasiert. Watson könnte aber auch auf Grundlage von Metriken oder schon gewonnen Informationen agieren. Dadurch könnte Watson versuchen mehr oder prisantere Informationen zu erhalten. Außerdem könnte Watson auch Spiel übergreifend Interagieren. Dadurch wäre auch das ableiten von Informationen möglich, wie zum Beispiel, wer spielt mit wem. Ist es wahrscheinlich das sich diese Spieler auch außerhalb des Spiels kennen. Wenn ja, könnte Watson einem der beiden Deals anbieten der beiden Spieler Hilft oder mit absicht dem anderem Spieler schadet.

7.2 Persistenz

Um den Prototypen sinnvoll für weitere wissenschaftliche Zwecke nutzten zu können, müssten die Spielverläufe persistiert werden. Der Prototyp hält momentan alle Spielverläufe im Arbeitsspeicher. Dies müsste um eine Persistierung auf eine Datenbank oder etwas vergleichbares erweitert werden. Die Persistierung auf eine Datenbank würde den Vorteil bieten die Daten auch außerhalb des Prototyps zugänglich zu machen.

7.3 Auswertung

Nun stellen sich unterschiedlichste Fragen die mit Hilfe dieses Systems betrachtet werden können und sich dann auf unsere Gesellschafft übertragen lassen. Solche Fragestellung könnten wie folgt aussehen?

- Ab wann empfindet der Spieler ein 'Schummeln' als legal/plausibel oder unrechtmäßig?
- Wie regieren Gewinner des Spiels auf Belobung/Tadel?
- Ändert sich die Einstellung der Spieler gegenüber dem Spiel abhängig vom Spielerfolg?
- Inwiefern verändert sich die Bereitschaft zur Informationspreisgabe über den Verlauf des Spieles?
- Bis zu welchem Grad ist ein Spieler bereit, bewusst zum Nachteil anderen Spieler zu agieren?

Um diese Fragen besser beantworten zu können, müssten die Aufgezeichneten Spieldaten analysiert und aufbereitet werden. Zusätzlich müsste das Spiel um einen Feedback bzw. Fragemechanismus erweitert werden, der dem Spieler direkt nach dem Spiel

Fragen über das Spielerlebniss stellt und diese mit den Daten über den Spielverlauf in Verbindung bringt.

References

2018. Valve removes scam game from steam, bans developer. *IGN*. https://www.ign.com/articles/2018/07/31/valve-removes-scam-game-from-steam-bans-developer.

2018. China's social credit system 'could interfere in other nations' sovereignty'. *The Guardian*. https://www.theguardian.com/world/2018/jun/28/chinas-social-credit-system-could-interfere-in-other-nations-sovereignty.

2018. Fake-likes. *Facebook*. https://de-de.facebook.com/business/a/page/fake-likes.

2015. Ex-Fifa-Funktionär Blazer gesteht Korruption bei WM 1998 und 2010. *Frankfurter Allgemeine Zeitung GMBH*. http://www.faz.net/aktuell/sport/sportpolitik/wm-1998-und-2010-ex-fifa-funktionaer-blazer-gesteht-korruption-bei-wm-1998-und-2010-13628556.html.

2018. China wirft Interpol-Chef Korruption vor. *NTV*. https://www.n-tv.de/politik/China-wirft-Interpol-Chef-Korruption-vor-article20659271.html.