CODE IS INTERPRETATION – LEGAL EXPLAINABILITY UND SOFTWARE-ENTWICKLUNG

Felix Gantner / Johannes Gärtner

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

XIMES GmbH calcuLex GmbH
Hollandstr 12/12, 1020 Wien AT
gaertner@ximes.com; http://www.ximes.at

Schlagworte: ***Transparenz, Legal Tech, Legal Decision Support Systems, Software-Entwicklung, Explainability, automatisierte juristische Begründungserzeugung, Arbeitszeitrechner***

Abstract: *Begründungen stellen einen wesentlichen Bestandteil juristischer Entscheidungen dar. Sie sichern deren Überprüfbarkeit. Transparenz und Nachvollziehbarkeit sind Voraussetzungen für den Einsatz von Systemen zur Unterstützung juristischer Entscheidungen. Das bedeutet, dass diese Systeme für einen Juristen die Prämissen und einzelnen Schritte der getroffenen Schlussfolgerungen nachvollziehbar machen. In der technischen Umsetzung führt dies zu einer doppelten Abbildung von juristischen Strukturen: einerseits als Programmcode als Sprache der Softwareentwickler, anderseits als Begründungselement in der Sprache des Juristen. Verbunden werden diese beiden Sichtweisen auf den Code durch Tests auf den unterschiedlichen Systemebenen. Am Beispiel des Arbeitszeitrechners wird die praktische Umsetzung dargestellt.*

1. Juristische Begründungen

Ein zentrales Element jeder juristischen Entscheidung, egal ob Urteil oder Bescheid[1], ist die Begründung. Die Begründung hat eine besondere Stellung, weil juristische Aussagen nicht durch z.B. Experimente überprüft und verifiziert bzw. zumindest falsifiziert werden können. Es ist nur möglich, über die Begründung die logische Folgerichtigkeit und Widerspruchsfreiheit einer Argumentation zu überprüfen und an Hand der in der Begründung enthaltenen Bewertung der Entscheidungsprämissen die getroffenen Wertungen nachzuvollziehen.

Ohne die Begründung bleibt «lediglich der *faktische* Glaube an die Richtigkeit und Werthaltigkeit bestimmten Sollens»[2].

Man kann daher Neumann folgen, wenn er feststellt: «Rechtswissenschaftliche Theorien können nicht verifiziert werden, sondern sie können und müssen *begründet* werden.»[3] Unter dem Begriff «rechtswissenschaftliche Theorien» sind heute nicht mehr nur Aussagen in Urteilen, Bescheiden oder juristischen Fachpublikationen zu verstehen. Vielmehr ist darunter durch die technische Entwicklung auch Software für juristische Anwendungsbereiche einzuordnen. Wann immer juristische Fragestellungen in Computerprogrammen abgebildet werden, ist das Programm eine Umsetzung einer «rechtswissenschaftlichen Theorie».

Betrifft die Software nur die Organisation oder Strukturierung rechtlicher Abläufe, wie z.B. ein elektronisches Aktensystem, so sind die Fragen, die die «rechtswissenschaftliche Theoriebildung» betreffen, überschau- und als internes Problem der Organisation – der Behörde, des Gerichts, … – behandelbar.

---

1 Vgl. § 60 AVG: «In der Begründung sind die Ergebnisse des Ermittlungsverfahrens, die bei der Beweiswürdigung maßgebenden Erwägungen und die darauf gestützte Beurteilung der Rechtsfrage klar und übersichtlich zusammenzufassen.»

2 Luhmann, Niklas, Legitimation durch Verfahren, S. 239 (Hervorhebung im Original).

3 Neumann, Ulf, Wissenschaftstheorie der Rechtswissenschaft, S. 382 (Hervorhebung im Original).

Sollen die Systeme jedoch in das «Kerngeschäft» von Juristen eingreifen, sie also als Legal Decision Support-Systeme die «rechtswissenschaftliche Theoriebildung» eines Richters, … unterstützen und damit beeinflussen, oder wird in Zukunft mancher KI-Anwendung noch weitergehende Entscheidungsmöglichkeit oder -autonomie zugestanden, dann ändert sich die Struktur juristischer Entscheidungsfindung wesentlich.[4]

Es ist daher nicht verwunderlich, dass seit den Anfängen[5] der Beschäftigung mit juristischen Expertensystemen die Frage der bzw. die Forderung nach Transparenz und Nachvollziehbarkeit einen hohen Stellenwert hat. Dies hat sich bis heute nicht geändert:

> Als wesentliches Element von «accountability» wird es angesehen, dass das Handeln einer Instanz erklärbar sein muss. Dieser Anforderung ist dann nicht Genüge getan, wenn der verwendete Algorithmus als bloße «black box» erscheint und einer Erklärung nicht zugängliche Resultate produziert, selbst wenn diese – für sich betrachtet – als korrekt erscheinen mögen.[6]

2. Theorie juristischer Begründungen

An den Rechtstheorie wird bei der Untersuchung der Struktur juristischer Begründungen zwischen interner und externen Rechtfertigung unterschieden[7].Die interne Rechtfertigung betrifft die Frage, ob ein juristisches Urteil aus den in der Begründung angegebenen Prämissen logisch folgt. Bei der externen Rechtfertigung geht es darum, ob die angegebenen Prämissen richtig sind.

Auf die logischen Modelle der Rechtsanwendung umgelegt, bedeutet diese Unterscheidung, dass zwischen der logischen Abbildung juristischer Argumentationsschritte – egal ob als Justizsyllogismus bzw. juristischer Syllogismus[8] oder im Rahmen der Softwareentwicklung in einer formalen Programmiersprache – und den Variablenwerten in dieser Darstellung unterschieden werden muss. Die Prüfung der in die Variablen eingesetzten Werte ist die externe Rechtfertigung, die der einzelnen logischen Ableitungsschritte die interne Rechtfertigung.

So könnte eine Abbildung von § 3 Abs. 1 Arbeitszeitgesetz

§ 3. (1) Die tägliche Normalarbeitszeit darf acht Stunden, die wöchentliche Normalarbeitszeit vierzig Stunden nicht überschreiten, soweit im Folgenden nicht anderes bestimmt wird.

in einer Programmiersprache lauten:

```
boolean validiereNormalArbeitsZeit(float tagesArbeitsZeit, float wochenArbeitszeit) { return (tagesArbeitsZeit <= 8) && (wochenArbeitszeit <= 40);}
```

In dieser Darstellung wird die Grenze zwischen interner und externer Rechtfertigung deutlich. Die Eingangsparameter tagesArbeitsZeit und wochenArbeitsZeit sind die Prämissen für die Funktion und werden im Rahmen der externen Rechtfertigung geprüft. Die Funktion validiereNormalarbeitsZeit, die die die logische Struktur der Rechtsnorm abbildet, prüft die Parameter nicht mehr inhaltlich, sondern geht von deren Korrektheit aus. Ob die danach folgende Bearbeitung und Auswertung der übergebenen Werte korrekt ist, ist der Inhalt der internen Rechtfertigung.

Externe und interne Rechtfertigung stellen zusammen die Voraussetzung für die Nachvollziehbarkeit und damit auch für die Überprüfbarkeit von juristischen Entscheidungen dar.

---

4 Vgl. dazu O'Neil, Cathy, Angriff der Algorithmen, Kapitel 5 «Zivile Opfer», S. 117ff.

5 Bund, Elmar, Einführung in die Rechtsinformatik, S. 303: «Will man juristische Expertensysteme zur Unterstützung von Organen der Rechtspflege einsetzen, so ist wegen deren persönlicher Verantwortung auf Transparenz und Nachvollziehbarkeit aller Verarbeitungsschritte großer Wert zu legen. Die Erklärungskomponente muß also sehr leistungsfähig ausgestaltet werden.»

6 Herberger, Maximilian, «Künstliche Intelligenz» und Recht, S. 2828. Vgl. auch Waltl, Bernhard/Roland Vogl, Explainable Artificial Intelligence – The New Frontier in Legal Informatics.

7 Alexy, Robert, Theorie der juristischen Argumentation, S. 273.

8 Pavcnik, Marijan/Lachmayer Friedrich, Der «juristische Syllogismus» als rationaler Rahmen der Entscheidung und seine Anwendung im elektronischen Formularverfahren.

Das beschriebene rechtstheoretische Modell beschreibt aber nur einen Teil der Argumentationsschritte, die für ein juristisches Urteil notwendig sind. Bei der Formulierung des Justizsyllogismus wird meist davon ausgegangen, dass die anzuwendende Norm bereits bekannt ist und nur mehr die Argumente, die bei der Anwendung der konkreten Norm auf einen bestimmten Sachverhalt vorgebracht werden, der Rechtfertigung bedürfen.

Tatsächlich steht man aber bei der Rechtsanwendung jedoch häufig vor dem Problem, dass die Anwendung mehrerer Rechtsnormen denkbar wäre, dass diese u.U. in einem Regel-Ausnahme-Verhältnis zu einander stehen, und dass die Auswahl der tatsächlich anzuwendenden Rechtsnorm ein begründet werden und nachvollziehbar sein muss. Diese Argumentationsschritte werden im Folgenden als systematische Rechtfertigung bezeichnet.

Bei der systematischen Rechtfertigung spielen juristische Formulare, zu denen auch Benutzerschnittstellen von juristischer Software bzw. Anwendungen gehören, eine besondere Rolle, da in Formularen die Sachverhaltsmerkmale mehrere Rechtsnormen zusammengefasst[9] werden können. Die über das Formular erfassten Daten liefern die Grundlage für Auswahl der tatsächlich anzuwendenden Rechtsnormen und sind damit Teil der systematischen Rechtfertigung.

Legal explainability von Software, die rechtliche Fragestellungen abbildet und damit eine Interpretation von Rechtsnomen darstellt, muss daher die Anforderungen der systematischen, externen und internen Rechtfertigung erfüllen, d.h. sie muss nachvollziehbar «erklären» und rechtfertigen, wie die Rechtsvorschriften interpretiert werden und damit ein konkretes Ergebnis berechnet wird.

3. Abbildung von Begründungsstrukturen in Software

Bei der Frage der *legal explainability* sind zwei Bereiche zu unterscheiden. Es ist dies einerseits die Begründung der Ergebnisse einer Beurteilung eines Einzelfalls bei Verarbeitung der Daten, die diesen Fall beschreiben. Und andererseits stellt der Prozess der Softwareentwicklung, bei dem Rechtsnormen bzw. Teile eines Rechtsgebiets generell interpretiert und abgebildet werden, zusätzliche Anforderungen an die Beschreibung des Systemverhaltens.

Besondere Bedeutung haben für die *legal explainability* in Normen enthaltene Rechenregeln, sowie Zeit-, Termin- oder Fristabhängigkeiten, bei denen die zeitliche Abhängigkeit direkten Einfluss auf das Ergebnis der juristischen Sachverhaltsbewertung hat. In diesen Fällen ist das Ergebnis, das das System erzeugt, für den Benutzer noch schwerer nachvollziehbar als in «juristischen Standardsubsumtionen». Hinweise auf die bei der Berechnung angewendeten Algorithmen und auf die Zeitabhängigkeiten sind dann ein unabdingbarer Teil der ausgegebenen Rechtfertigung.

In beiden Bereichen ist zu unterscheiden, wer Adressat der Erklärung ist und dass Techniker, Juristen und der Anwender, wenn er nicht den beiden genannten Personengruppen angehört, nicht dieselbe Fachsprache sprechen und auch andere Anforderungen an die «Funktionsfähigkeit» des Systems haben.

3.1. Einzelfallbegründung

Eine Einzelfallbegründung ist jene Begründung, die das System erzeugt, wenn es ein Anwender benutzt und einen Sachverhalt bewerten lässt.

Für die Erfassung des Sachverhalts spielt die Benutzeroberfläche, das Formular eine zentrale Rolle. Ein wesentlicher Anteil der *legal explainability* ist aus Benutzersicht das Wissen, welche Daten an welcher Stelle eingetragen werden müssen und welche juristische Bedeutung ihnen dabei zugewiesen wird.

Diese oft unterschätzte Anforderung an die Eingabemasken oder die Schnittstellen geht inhaltlich über ein Hilfesystem, das die Benutzung eines Programms erklärt, hinaus. Aus Sicht der juristischen Rechtfertigungstheorie hat sie große Bedeutung für externe und die systematische Rechtfertigung.

[9] Gantner, Felix, Theorie der juristischen Formulare, S. 100.

- systematische Rechtfertigung: Durch die Eingaben in der Schnittstelle wird die Auswahl getroffen, welche von den zahlreichen Rechtsnormen, die in dem Formular überlagert dargestellt werden, auf den konkreten Sachverhalt, angewendet werden soll. Unabhängig davon, ob es dem Benutzer bewusst ist, steuert dieser durch seine Eingaben den nachfolgenden automatisierten Subsumtionsprozess. Das System muss ihm daher mitteilen, was für Daten es erwartet und welche Bedeutung diesen zugeordnet werden.
- externe Rechtfertigung: wie oben dargestellt, ist die Überprüfung der Richtigkeit den Daten, die als Parameter im juristischen Schluss (interne Rechtfertigung) verwendet werden, Teil der externen Rechtfertigung. Das System kann nur Plausibilitätsprüfungen vornehmen, nicht jedoch die inhaltliche Richtigkeit der eingetragenen Sachverhaltsmerkmale feststellen. So wird z.B. im Fachgebiet der Arbeitszeiten im Eingabebereich eine formale Prüfung durch das System auf negative Arbeitszeiten möglich und sinnvoll sein, die sonstige Korrektheit der Eingaben aber durch den Benutzer sichergestellt werden müssen.

Legal explainability für die Einzelfallbegründung bedeutet, dass das Ergebnis der juristischen Bewertung hat alle drei Bereiche der Rechtfertigung abzubilden und zu enthalten. Es müssen daher die eingegebenen Daten (externe und systematische Rechtfertigung) in Kombination mit den juristisch relevanten Klassifikationen durch die Software im Rahmen der Berechnungen dargestellt werden. Zusätzlich muss das Ergebnis der automatisierten Subsumtion (interne Rechtfertigung) und u.U. auch von Zwischenschritten so ausgegeben werden, dass es der Benutzer selbst als Fachexperte oder nach Konsultation eines juristischen bzw. Domänenexperten nachvollziehen kann.

3.2. Softwareentwicklung und Begründung

Bei der Entwicklung juristischer Softwaresysteme stellt sich das Problem, dass juristische auf technische bzw. formale Begriffs- und Denkstrukturen abgebildet und übertragen werden müssen. Dieser Vorgang ist die Voraussetzung dafür, dass das System überhaupt entwickelt werden kann.

Dabei stellt sich das Problem, dass sowohl Sprache, als auch Kategorienbildung und Begriffsstrukturen schwer in Einklang zu bringen sind. *Legal explainability* bei der Softwareentwicklung bedeutet zu Beginn in erster Linie, eine gemeinsame Sprache und ein gemeinsames Verständnis der Domäneninhalte zu finden.

Bei der Softwareentwicklung stellt sich meist als besonderes Problem heraus, dass zwar im ersten Schritt des Entwicklungsprozesses für die Abbildung genereller Rechtsnormen aus Gesetzestexten Juristen und Techniker eine gemeinsame logische Darstellung für die Strukturen und Begriffe finden. Sollen aber Begriffsinhalte z.B. in Klassenstrukturen übertragen werden, dann führen die unterschiedlichen methodischen Ansätze zu Problemen. Der Informatiker versucht möglichst umfassend und vollständig die Grundlagen der Begriffsstrukturen zu definieren, indem er zuerst die allgemeinen Eigenschaften der einzelnen Begriffe, Kategorien, … definiert und danach die Spezialfälle dazu implementiert. Die juristische Methode hingegen misst dem Standardfall, dem Begriffskern, weniger Bedeutung zu, da diese ja «offensichtlich» ist. Es steht der strittige Einzelfall im Vordergrund, über den durch Abgrenzung die Begriffsinhalte definiert werden.

Der Programmcode als Interpretation von Rechtsvorschriften ist sowohl Übersetzung, als auch Übertragung von Strukturen.

Legal explainability im Rahmen der Softwareentwicklung bedeutet daher, den Transfer von juristischem Wissen zu den Entwicklern zu meistern. Dieser Transfer ist die Grundlage für die *legal explainability* bei der Einzelfallbegründung.

Im Rahmen der Systementwicklung können folgende Bereiche unterschieden werden:

- Abbildung im Code: Es wird versucht, die zu berücksichtigenden Sachverhalte zu identifizieren und darauf aufbauend Begriffe und logische Strukturen bzw. Zusammenhänge aus den Rechtsnormen im Programmcode abzubilden. Bei testgetriebener Entwicklung können Unit-Tests zur Dokumentation juristischer Zusammenhänge oder Eigenschaften genutzt werden. Diese Tests dienen der Sicherung der

technischen Korrektheit und Konsistenz, unterstützen aber auch die *legal explainability*, da ja auch auf der Ebene des Programmcodes juristische Transparenz durch die Zuordnung von Textelementen der zu generierenden Begründungen erzeugt wird.
- Integration: Auf dieser Ebene der Entwicklung kann das juristische Wissen und Können der Interpretation von Normen an Hand von Einzelfällen genutzt werden, um das Zusammenspiel der einzelnen Programmteile zu überprüfen. Übergreifende juristische Begründungszusammenhänge und Abgrenzungen werden über Einzelfälle und daraus gewonnene Integrationstest dargestellt.
- Systemtests, wenn möglich automatisiert, dienen der Sicherstellung und Dokumentation der korrekten Funktion des Systems. Änderungen der Rechtslage können durch Anpassungen der Tests geprüft werden. Das System zeigt über die Tests an, welcher Code von juristischen Änderungen betroffen ist.

Im Bereich der Softwareentwicklung hat *legal explainability* damit eine vorrangig technische Bedeutung und unterstützt die Entwicklung und Wartung des Systems. Die technische Darstellung und Beschreibung der juristischen Zusammenhänge ist die Grundlage für die *legal explaina*bility bei der Einzelfallabwendung.

4. Praxisbeispiel Arbeitszeitrechner für die AK-Wien

In einem interdisziplinären Entwicklungsprojekt wurde der «Arbeitszeitrechner» entwickelt, der von der AK-Wien auch in der Beratung verwendet wird. In der gegenwärtigen Ausbaustufe wurden Teile des Arbeitszeitgesetzes und des KV-Reinigung als Software implementiert. KundInnen der AK bringen ihre Arbeitszeitaufzeichnungen mit und es wird geprüft, ob die Zeiten richtig gerechnet wurden. Weiters wird ein entsprechender Bericht erzeugt, der dann für die weiteren Schritte die Ergebnisse nachvollziehbar zur Verfügung stellt.

Im Zuge der Entwicklung waren eine Reihe von rechtlichen Themen zu bearbeiten und von technischen Fragen zu lösen, die im Folgenden beschrieben werden. Z.T. wurden dazu eigenständige Werkzeuge entwickelt um Technik und Rechtsinterpretation besser zu verzahnen. Die entwickelten Methoden und Werkzeuge werden inzwischen auch von anderen Kunden genutzt.

4.1. Abbildung von Rechtsinterpretation in Code mit dem calcuLektor

Wie oben beschrieben, stehen am Anfang der Nutzung von Software die Sachverhalte und darauf aufbauende Begriffe im Zentrum der Arbeit. Der von uns gewählte Ansatz versucht dabei drei Elemente der Interpretationsarbeit zu integrieren. Dies soll am Beispiel des «Arbeitstages» im Sinne des Arbeitszeitrechts gezeigt werden:

1. Code ist erforderlich aber schwer lesbar: Ein Codeelement zur Prüfung, ob ein neuer Arbeitszeittag begonnen hat, hat selbst in Excel-Notation Elemente, die ohne Kenntnisse in Programmierung nicht unmittelbar oder zumindest nur schwer zugänglich sind.
   WENN(Ruhezeit_Stunden>=11;Wahr;Falsch)
   WENN(UND(DauerInStunden(BeginnErstesIntervall;EndeZweitesIntervall)>24;Ruhezeit_Erfüllt=Falsch);Wahr;Falsch)
2. Um besser nachvollziehbar zu machen, was passiert *(explainability)*, werden dynamischen Erklärungstexte zur jeweiligen Berechnung bereits an dieser Stelle mitentwickelt. Damit wird sichtbarer, welche Interpretation von Programmierseite verwendet wurde. Diese ist dann sowohl zur Prüfung durch Juristen als auch dann für das Nachvollziehen durch Dritte hilfreich. Z.B. entstehen derartige Texte.
   Da keine ausreichende Ruhezeit vorliegt, verlängert sich der Arbeitszeittag.
   Der Zeitraum von Beginn der ersten Arbeit bis Ende des zweiten Blockes würde 24-Stunden überschreiten. Es beginnt daher nach 24-Stunden ein neuer Arbeitszeittag. Es liegt eine Ruhezeitverletzung vor.
3. Um besser nachvollziehbar zu machen, welche juristische Überlegungen hinter dieser Interpretation stehen, werden auch diese direkt an der Stelle dokumentiert. Z.B. der Verweis auf eine ausführliche Diskussion dieses Themas in:

Heilegger & Klein. AZG. 4. Auf. S179
Idealerweise nicht nur als Referenz, sondern so, dass der Auszug oder die Diskussion dazu komplett zur Verfügung stehen.

Diese drei Elemente werden von klassischen Programmierumgebungen, die viele Entwicklungsaufgaben unterstützen, so nicht direkt unterstützt. Um das Editieren in diesen drei Facetten von Anfang an zu erleichtern, entwickelten wir einen speziellen Editor («calcuLektor»), der diese und weitere Aufgaben (z.B. eine einfache Erzeugung kleiner Tests) erheblich leichter macht.

4.2. Integration

Die Integration der Rechtslogik in ein Gesamtsystem, mit dem Fälle bearbeitet werden können, ist wie oben dargestellt eine eigene, komplexe Aufgabenstellung. Zum Teil geht es um klassische IT-Themen, wie z.B. Datenhaltung. Besondere Bedeutung hatten im Beispielprojekt große Fallbeispiele, die von JuristInnen konstruiert wurden, um Grenzfragen zu prüfen (z.B. wird auf Grund der gewählten Fristen ein Zuschlag wirksam oder nicht, entsteht Zeitausgleich oder nicht).

Zusätzlich treten aber auch viele praktische Fragen auf, die eng mit dem Recht verwoben sind und zusätzliche Interpretationen erforderlich machen können. Einige Beispiele.

- **Umgang mit unterschiedlich dargestellten Sachverhalten:** Im Allgemeinen ist von unterschiedlicher Darstellung von Sachverhalten und unterschiedlicher Verfügbarkeit ausgehen. So auch beim Arbeitszeitrechner. Hier musste ein Umgang mit Arbeitszeitaufzeichnungen unterschiedlichster Art gefunden werden: z.B. Arbeitszeitaufzeichnungen mit konkreten Pausenzeiten und andere Aufzeichnungen, in denen nur die Länge in Minuten verfügbar war. Dabei war zu entscheiden, ob diese überhaupt gerechnet werden sollen (in vielen Fällen müsste eine Pause zeitlich bekannt sein) und ob sie in unterschiedlicher Rechtslogik gerechnet werden sollen (und können). Im konkreten Fall, fiel die Entscheidung Pausenzeiten einheitlich zu handhaben. Als faires Lösungs-Verfahren für fehlende Pausenzeiten wurden die Pausen fiktiv in die Mitte der Arbeitszeit gelegt und als juristisch so vertretbar beurteilt.
- **Umgang mit komplexen Sachverhalten:** Nicht immer ist der Sachverhalt unmittelbar klar und es braucht z.T. juristisches Vorwissen oder zumindest Checks an dieser Stelle. Beim Arbeitszeitrechner war hier die Frage des Umgangs mit Abweichungen von der vereinbarten Lage der Normalarbeitszeit ein großes Thema. Z.B. sind bei werktäglicher Abweichung Abwesenheitszeiten zu unterscheiden, die zu vergüten sind (z.B. Urlaub) oder anders vereinbart wurden (z.B. Vereinbarung einer anderen Lage). Ähnliches gilt für Zusatzzeiten (z.B. Überstunde oder nur veränderte Lage). Hier musste eine praktikable Lösung gefunden werden, die vermeidet Kunden zur Zuordnung von Zeiten zu zwingen, wenn Ihnen die Regeln dazu vielleicht nicht bekannt sind. Gleichzeitig soll es für FachreferentInnen einfach bleiben, diese Sachverhalte zu sichten und gegebenenfalls zu ergänzen / zu ändern (und dies in nachvollziehbarer Weise). Dazu braucht es eigene Eingabe- und Interaktionsmöglichkeiten und rechtliche Einschätzung was wie vertretbar/sinnvoll ist.
- **Umgang mit Defaultwerten:** Defaultwerte haben oft eine höhere Wahrscheinlichkeit verwendet zu werden. Einerseits weil dafür weniger Arbeit erforderlich ist, andererseits werden ja auch der Defaultwert meist bewusst so gewählt, um bei Unsicherheit möglichst einen gangbaren Weg zur Verfügung zu stellen (und dies ist NutzerInnen wiederum bekannt). Zusätzlich ist es für NutzerInnen oft leichter, einmal ein Ergebnis zu sehen und dann gegebenenfalls anzupassen, statt sofort etwas eingeben zu müssen. Im Projekt Arbeitszeitrechner stellte sich damit die Frage, ob es juristisch vertretbar ist, anzunehmen, dass z.B. Zeiten normalerweise verschoben werden oder wären stattdessenn Überstunden als Standard besser oder sollte jeder Einstellung geprüft werden, oder ein fallspezifischer Defaultwert besser wäre. Hier geht es also auch um Positionierung der Interpretation und die Frage was wie gut rechtlich vertretbar ist.

- **Wording:** Während die Bezeichnungen bestimmter Sachverhalte und Ergebnisse auf der Ebene der einzelnen Units noch eher unkritisch war, spielte es hier eine zentrale Rolle. Dabei wurde es sichtbar, dass es nicht nur um Interpretation des Rechts, sondern auch die Ergebnisse wiederum interpretiert werden und es hier sehr leicht zu Missverständnissen und Unsicherheiten kommt.
- **Prüfung auf Unzulässigkeit oder Unklarheit:** Bestimmte Sachverhaltseingaben sind technisch oder für Spezialisten zum Thema leicht als unzulässig zu erkennen (z.B. überlappende Arbeitszeiteingaben) bzw. waren juristisch zu prüfen (z.B. ist es möglich einen Zeitausgleich zu nehmen, bevor ein Zeitguthaben erworben wurde). Manchmal sind bestimmte Angaben nur in bestimmten Konstellationen erforderlich (z.B. die genaue Lage eines Zeitabbaus, wenn davon die Einhaltung einer Frist abhängt). Um unnötige Arbeiten zu ersparen, kann dann meist auf Eingaben verzichtet werden, in manchen Fällen muss aber sichergestellt werden, dass die erforderlichen Zusatzeingaben oder fehlerhafte Eingaben nicht durchrutschen. Damit mischen sich Rechtsinterpretationsfragen («das darf nicht so gerechnet werden.») mit Usability Themen.

4.3. Systemtests

Die großen konstruierten Integrationstests wurden als Systemtests weiter genutzt. Sowohl für die Systemtests als auch für die Nachnutzung der Ergebnisse entstanden dabei zusätzlich Anforderungen an die Aufbereitung. Diese zielten z.T. auf Vereinfachung der Nutzung ab (z.B. Inhaltsverzeichnisse, Zusammenfassung bestimmter Kontenstände zum Ende). Die Berichte wurden auch auf Grund der Erklärungen umfangreich (z.B. 50 Seiten), z.T. zielten sie aber auch auf sichtbarmachen bestimmter Verläufe (z.B. Entwicklung bestimmter Salden im Zeitverlauf), was für die Prüfung der Korrektheit sehr hilfreich war.

Besonders profitierte das Projekt von der Entwicklung eines eigenen Differenzberichts, mit dem nicht nur geprüft werden konnte, ob sich von Version zu Version etwas geändert hatte, sondern auch mitgeliefert wurde, wie viele und welche Änderungen es wo gab. Die Nützlichkeit ergibt sich aus dem großen Volumen eines Tests (oft 5–10.000 zu prüfende Zahlen und zu prüfende Texte für einen Testfall).

5. Ausblick

Wenn Code eine konkrete Interpretation des Rechts ist – was dieser Artikel zu zeigen versuchte -, dann ist es zentral, dass er in die externen Rechts-Begründungen transparent eingebettet ist und die interne Rechtsinterpretation sichtbar macht. Die hier entwickelten Ansätze und Methoden unterstützen dies zum Teil bereits (z.B. calcuLektor mit dem Zusammenspiel von Code, Erklärung und Begründung). In anderen Bereichen wurden die Problemstellungen klarer (z.B. Umgang mit unterschiedlich, unvollständigen, komplexen … Sachverhalten). Spezialisierte Werkzeuge zur Entwicklung der Rechtsinterpretationen und ihrer Nutzung werden damit zentral.

6. Literatur

Alexy, Robert, Theorie der juristischen Argumentation, 2. Aufl., suhrkamp taschenbuch wissenschaft, Frankfurt am Main 1991.

Bund, Elmar, Einführung in die Rechtsinformatik, Springer-Verlag, Berlin Heidelberg New York Tokyo 1991.

Gantner, Felix, Theorie der juristischen Formulare, Duncker & Humblot, Berlin 2010.

Heilegger, Gerda, Klein, ChristophArbeitszeitgesetz, 4. Aufl., ÖGB Verlag, Wien 2016.

Herberger, Maximilian, «Künstliche Intelligenz» und Recht, NJW 39/2018, S. 2826.

Luhmann, Niklas, Legitimation durch Verfahren, 10. Aufl., suhrkamp taschenbuch wissenschaft, Frankfurt am Main 2017.

Neumann, Ulf, Wissenschaftstheorie der Rechtswissenschaft, in: Kaufmann, Arthur/Hassemer, Winfried (Hrsg.), Einführung in Rechtsphilosophie und Rechtstheorie der Gegenwart, 5. Aufl., C. F. Müller Juristischer Verlag, Heidelberg 1989, S. 375–390.

O'Neil, Cathy, Angriff der Algorithmen, Carl Hanser Verlag, München 2017.

Pavcnik, Marijan/Lachmayer Friedrich, Der «juristische Syllogismus» als rationaler Rahmen der Entscheidung und seine Anwendung im elektronischen Formularverfahren, in: Schweighofer, Erich/Kummer, Franz/Hötzendorfer, Walter/Borges, Georg (Hrsg.), Netzwerke/Networks Tagungsband des 19. Internationalen Rechtsinformatik Symposions IRIS 2016, Österrreichische Computer Gesellschaft & Erich Schweighofer, Wien 2016, S. 319–325.

Waltl, Bernhard/Roland Vogl, Explainable Artificial Intelligence – The New Frontier in Legal Informatics, in: Schweighofer, Erich/Kummer, Franz/Saarenpää, Athi/Schafer, Burkhard (Hrsg.), Datenschutz/Legal Tech Tagungsband des 21. Internationalen Rechtsinformatik Symposions IRIS 2018, Editions Weblaw, Bern 2018, S. 113–122.