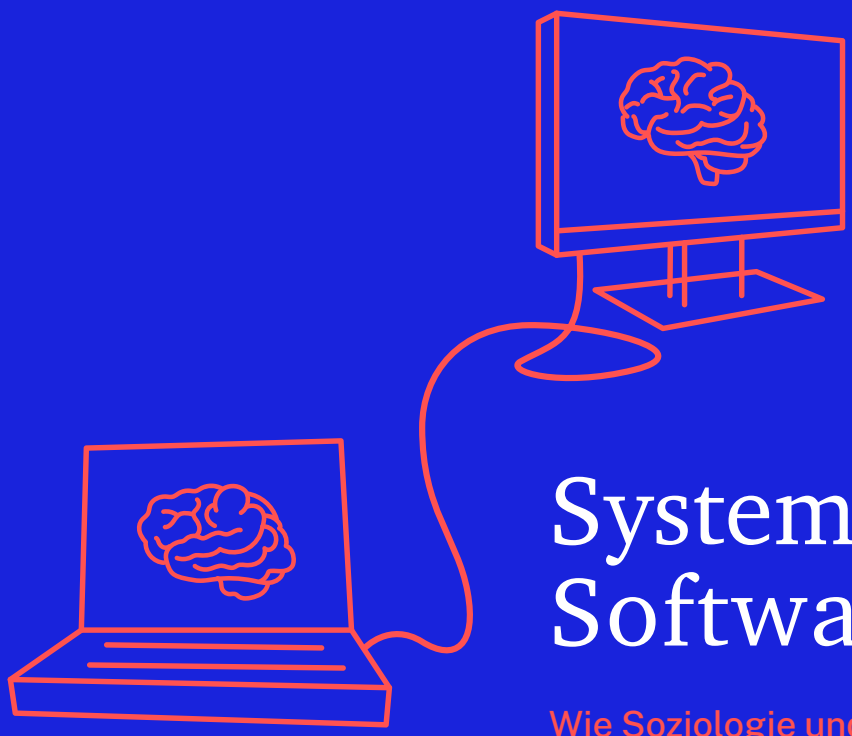

Systemtheorie & Softwaredesign

Wie Soziologie und Softwareentwicklung zusammenspielen können

Heutige Softwareentwicklung wendet häufig Techniken des modernen Domain-Driven Design an. In dessen Kern gilt es, mit geeigneten Kommunikationsmitteln abgeschlossene Kontexte zu finden und in Software zu gießen (Achtung: Conway's Law). Die soziologische Systemtheorie nach Luhmann fokussiert auf Kommunikation und nimmt in ihrem Kern die Unmöglichkeit unabhängigen Beobachtens an. Zusammengenommen kann dies ein besseres Verständnis für die Anforderungsfindung und das Zusammenstellen und -arbeiten von Teams liefern.

Ein Artikel von Dr. Christian Mennerich & Frederick Meseck

Dr. Christian Mennerich (mennerich@synyx.de) arbeitet seit 2013 für die synyx GmbH & Co. KG mit Sitz Karlsruhe. Christian beschäftigt sich unter anderem mit Datenmodellierung, noSQL und nachrichtenorientieren, verteilten Systemen. Er ist immer wieder als Sprecher auf Konferenzen unterwegs, schreibt Artikel und lektoriert gelegentlich. Seit 2022 ist er als Repräsentant von synyx in England, um mit der synyx UK Ltd. die Ideen und Werte von synyx auch im Vereinigten Königreich zu etablieren.

Wer Software nach modernen Paradigmen entwickelt, wendet oft Prinzipien des domänengetriebenen Softwaredesigns [Eva03] an. Zentral ist hier das Konzept der abgeschlossenen Kontexte. Ihre Struturen und Zusammenhänge haben großen Einfluss auf die Qualität des entstehenden Softwareprodukts, ihre Schnitte sind das Fundament für die Entwicklung guter und akzeptierter Softwareprodukte, denn hier spiegelt sich die Inter- und Intrakontextkommunikation wider.

Conway's Law [Con68] besagt nun, dass diese Kontexte eine Kopie der Kommunikationsstrukturen in der Organisation sein müssen (Isomorphieannahme). Das heißt auch, dass sich Conways Gesetz nur richtig interpretieren und nutzen lässt, wenn die Kommunikationsstrukturen verstanden sind, die das System kopiert. Deshalb ist bereits in der Anforderungsfindung und dem Requirements Engineering darauf zu achten, „relevante" Kommunikationsstrukturen zu entdecken und offenzulegen. Eine soziologische Theorie wie die Systemtheorie nach Niklas Luhmann und die darin enthaltenen konstruktivistischen Ansätze [Luh87] sind aus unserer Sicht keine esoterischen Gedankenspielereien, sondern können wertvolle Werkzeuge bei der Bewältigung dieser nicht trivialen Aufgabe sein.

Kontextabgrenzung

Das domänengetriebene Softwaredesign, wie es vor allem Evans [Eva03] und Vernon [Ver13] geprägt haben, findet heute in vielen Softwareprojekten Anwendung. Dabei steht das Geschäft, die Anwendungsdomäne, im Kern aller Betrachtungen. Im Zuge der Entwicklung von Software in modulithischen, Microservice- orientierten oder auch serverless Architekturen kommt hier auch die Betrachtung von Conway's Law und Varianten wieder verstärkt in die Diskussion (vgl. [Ske19], [For18]). Dieses verknüpft die Kommunikationsstrukturen im Unternehmen mit dem System, das diese abbildet, in strenger Interpretation isomorph [Euro PLoP05]. Isomorphie bedeutet strukturelle Ununterscheidbarkeit und ist in diesem Sinne in beide Richtungen zu verstehen:

- Die „Gesellschaft" der Anwender prägt das Design der Systemlandschaft,
- umgekehrt verändert diese aber wiederum auch das Verhalten der Anwender.

Beim Finden der „richtigen" Systemschnitte wird dabei mehrschichtig vorgegangen: Von der Organisation über das Finden von Unter- und Kernunterdomänen (Subdomains, Core(sub)domains) in einem Problemraum hin zur Implementierung abgeschlossener Kontexte (bounded contexts) in einem Lösungsraum.

Wir werden einen Vorschlag machen, wie die Systemtheorie nach Luhmann und insbesondere die darin enthaltenen konstruktivistischen Ansätze helfen können, die Kommunikationsstrukturen in der Organisation zu beschreiben und wichtige, sogenannte „relevante", Kommunikationsflüsse zu enthüllen. Diese, soziale Systeme genannten, Strukturen schlagen wir als Basis für die Abbildung auf abgeschlossene Kontexte vor. Die hohe Ähnlichkeit von Prinzipien des DDD (domain-driven design, [Eva03]) und der Systemtheorie sind bereits in ihrer verwendeten Sprache erkennbar. Aus dem Vorgehen resultiert eine Betrachtung der iterativen (und agilen) Entwicklungsprinzipien von einer anderen Warte aus.

In diesem Artikel begleiten wir einen Requirements Engineer (RE) bei der Anforderungsanalyse für ein digitales Transformationsprojekt im Bereich der Hinterlandcontainerlogistik. Dabei wird er drei verschiedene Herangehensweisen verwenden. Der RE beginnt mit einem *klassischen Ansatz* aus dem Requirements Engineering. Dieser wird angereichert mit Aspekten, die dem *Konstruktivismus* entlehnt sind. Und letztlich kommt eine *systemisch-agile Sichtweise* hinzu. Auf dem Weg diskutieren wir die Auswirkungen, die diese Sichtweisen auf Architektur, Qualität und Akzeptanz eines entstehenden Softwareprodukts haben können.

Anwendungsdomäne: Containerlogistik an Hinterlandterminals

Unsere Anwendungsdomäne ist die Containerhinterlandlogistik. Hier geht es im Kerngeschäft um den Transport von Waren in standardisierten Containern, entweder in die großen Überseehäfen (Exportfall) oder von dort ins Binnenland (Importfall). An Containerterminals werden Container auf verschiedene Modalitäten (hier LKW, Binnenschiff oder Bahn) umgeschlagen oder auch gelagert.
In Zukunft sollen möglichst viele Schritte (voll-)automatisch geschehen, damit die Containerterminals effizienter und sicherer arbeiten können. Dazu sind die relevanten Prozesse, Abläufe und Handlungen rund um den Containertransportauftrag zu erfassen, zu verstehen und letztlich in einer geeigneten Softwarearchitektur zu implementieren. Wir begleiten unseren RE dabei und schauen, wie sich seine Weltsicht auf seine Arbeit auswirkt.

Der Klassiker

Unser RE beobachtet an einem Hinterlandterminal folgendes Szenario: Ein Kunde beauftragt die Salesabteilung des Containertransportunternehmens mit dem Export seiner Ware nach Übersee. Somit entsteht ein Auftrag, der den Rahmen für alle weiteren Transportdienstleistungen vorgibt. Ein Disponent reserviert einen LKW, mit dem der Container im Rahmen des Auftrags transportiert werden soll. Dieser LKW fährt zu gegebenem Zeitpunkt mit einem leeren Container zum Ort der Beladung, wartet dort und fährt dann den beladenen Container wieder zurück zum Terminal. Am Gate (dem Zugangsort zum Terminal) werden Berechtigungen und Sicherheitsvorschriften geprüft. Ist alles wie vorgeschrieben, darf der LKW mit dem Container einfahren. Ein sogenannter Reachstacker hebt den nun vollen Container vom LKW und stellt ihn auf das Terminal. Später wird der Container das Terminal in Richtung Seehafen mit einem Binnenschiff oder Zug wieder verlassen. Ein typischer Aufbau eines Hinterlandterminals und der identifizierten Abteilungen ist schematisch in **Abbildung 1** gezeigt.

Der RE legt seinen Beobachtungen ein Unternehmensorganigramm zugrunde und sieht: Salesabteilung, Disposition und Gate, LKWs, Stacker und Kräne. Er beobachtet deren reale Interaktion, und schreibt formalisiert nieder, was er sieht. Die entstehenden Dokumentationen (sei es ein Anforderungsdokument oder ein

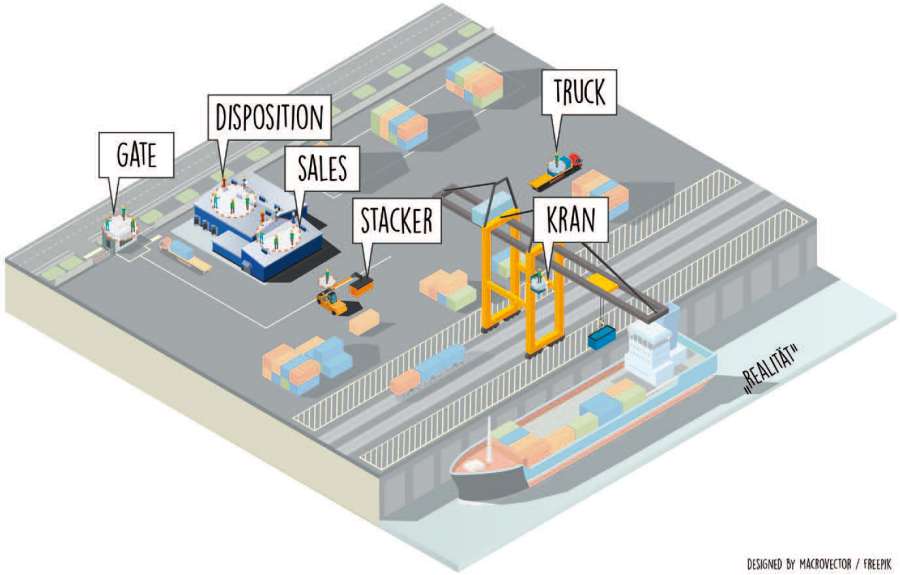

Abb. 1: Schematischer Aufbau eines Containerterminals, wie es ein Containerumschlagsunternehmen unterhalten könnte. Hervorgehoben sind einige Abteilungen, wie sie Teil eines Organigramms sein könnten, sowie einige Aktoren am Terminal.
Bilder: Designed by macrovector/Freepik, bearbeitet durch die Autoren

Backlog) werden an Softwareentwicklungsteams gegeben, die es entsprechend ihrem Verständnis in ein Softwareprodukt umsetzen. Dabei ist je ein Team für eine Struktur im Organigramm zuständig.

Häufig ist das nach längerer Entwicklungszeit entstehende Ergebnis ernüchternd, das oft unterschiedliche Verständnis der Domäne ist unmittelbar sichtbar (vgl. **Abbildung 2**). Schnittstellen zwischen Abteilungen und Teams sind sichtbar und passen nicht zueinander, Prozesse sind falsch oder unzureichend abgebildet. Die digitalisierte Realität und die erlebte Realität haben wenig gemein, die Software findet nur mäßige Akzeptanz.

Der Konstruktivist

Unser RE kommt nun zu der Annahme, dass Realität „an sich" gar nicht erfassbar und beschreibbar ist, sondern dass jede Beschreibung immer den Moment einer Wahrnehmung widerspiegelt. Damit kann auch das, was ein Team in Software umsetzt, nur das momentane Verständnis des Teams sein.

Unser RE sorgt jetzt dafür, dass sich alle Teilnehmer am Softwareentwicklungsprozess an einen Tisch setzen, und über die Domäne sprechen. Durch diese Kommunikation soll ein Realitätskonsens zwischen Entwicklerteam und Anwenderschaft über das zu entwickelnde Produkt entstehen, das heißt eine Einigkeit darüber, wie die nicht objektiv erfassbare Realität zu verstehen ist. Dies gelingt am besten in der direkten sprachlichen Kommunikation. Gesprochen wird über ein Modell, das in der konstruktivistischen Sichtweise zu verstehen ist als „eine beschränkte und ziel-

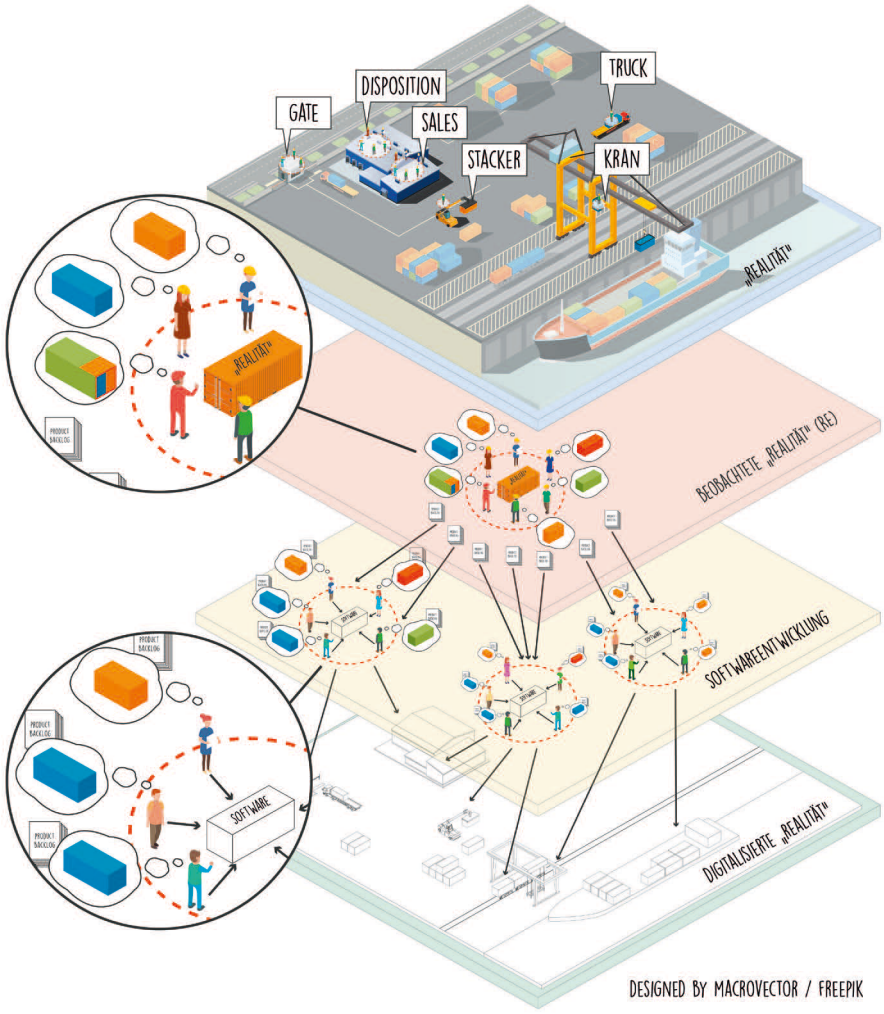

Abb. 2: Direkte Abbildung von Prozessen in Software in einem klassischen Ansatz. Unterschiedliches Verständnis der Domäne führt zu Missverständnissen in einem Softwareprodukt, das daraufhin nur mäßige Akzeptanz erfährt.
Bilder: Designed by macrovector/Freepik, bearbeitet durch die Autoren

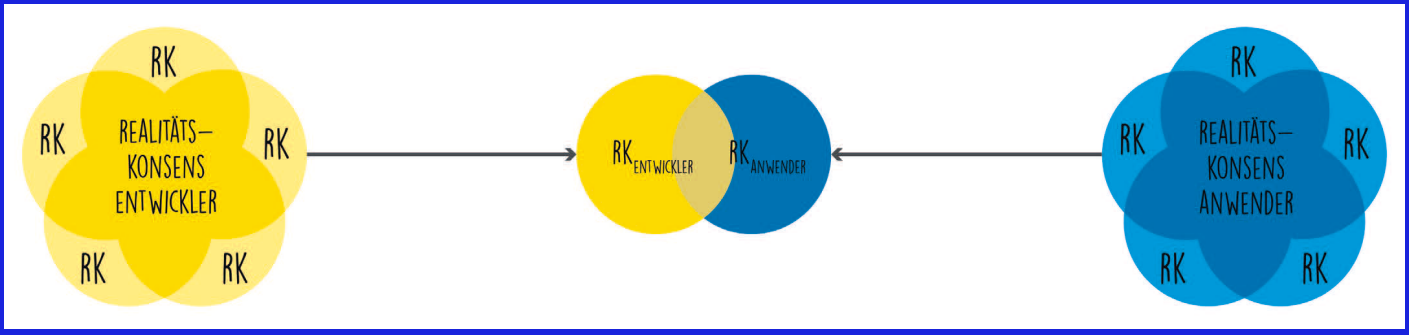

Abb. 3: Illustration, wie die Konsense zwischen Anwendern und Entwicklerteam gedacht werden können. Eine hohe Übereinstimmung hier zeugtvon einem guten gemeinsamen Verständnis der Anwendungsdomäne und davon, wie Software hier unterstützen kann. Dies wird höchstwahrscheinlich auch zu einer hohen Akzeptanz der Software führen.

gerichtete Beschreibung einer Realitätskonstruktion durch einen Beobachter"[1].

Im Entwicklungsprozess haben Anwender und Entwicklungsteam jeweils einen gemeinsamen Konsens. Diese Konsense gilt es, möglichst gut zur Deckung zu bringen (siehe **Abbildung 3**). Unser RE versteht aber, dass er teilhat am Konsens der Anwender: Durch seine Anforderungsanalyse wird er ein Teil ihrer Gesellschaft und beeinflusst diese unweigerlich.

Der agile Systemiker

Unser RE orientiert sich nun zusätzlich noch stärker an der sogenannten Systemtheorie nach Luhmann, einer soziologischen Theorie. Diese ist sehr abstrakt und allgemein gehalten, und kann hier im Detail nicht diskutiert werden (vgl. z. B. [Luh87], [Luh92], [Wil82]). Ein paar ihrer Kernelemente, die unserem RE gewinnbringend erscheinen, werden wir aber genauer beleuchten.

Unser RE wirft nun einen Blick in das Gatehaus, welches im Sinne der Systemtheorie nach Luhmann ein sogenanntes soziales System bildet. Dies sind abgeschlossene Systeme, die sich ausschließlich über die darin stattfindende Kommunikation definieren. Sie haben ein paar wichtige Eigenschaften: Soziale Systeme erschaffen sich selbst und wollen fortbestehen (Autopoiesis), Kommunikation im Inneren findet regel- und musterhaft statt, und sie grenzen sich von einem Außen (einer Umwelt) ab. In der Konsequenz kann von außen niemals direkt auf die Kommunikation im Inneren zugegriffen werden, und damit ist ein soziales System selbstorganisiert und autonom (siehe auch **Abbildung 4**).
Im konkreten Fall des Gatehauses beobachtet der RE Folgendes: Ein LKW kommt am Terminal an, steht aber nicht auf der Voranmeldungsliste. Ein Mitarbeiter fragt seinen Kollegen, was nun zu tun ist. Die Antwort lautet: Der LKW soll einfahren und abgehandelt werden. DerVorgang soll aber auf die gestrige Liste geschrieben werden, auf der er ursprünglich stand

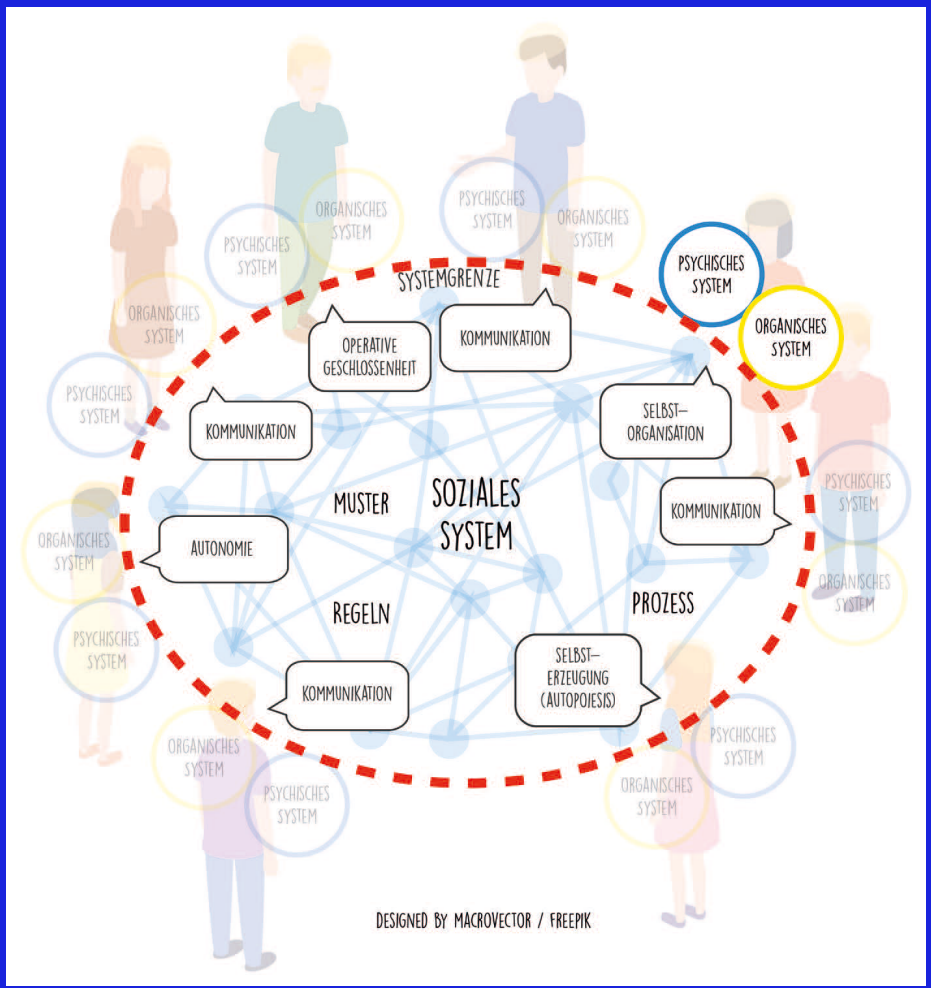

Abb. 4: Illustration, wie ein soziales System gedacht werden kann. Gezeigt sind auch sogenannte gekoppelte Systeme (hier physische und organische Systeme), die in unserer Betrachtung keine explizite Rolle spielen.
Bilder: Designed by macrovector/Freepik, bearbeitet durch die Autoren

[1] Im Gegensatz zur klassischen Modelldefinition als „eine beschränkte und zielgerichtete Beschreibung der Realität" [Wiki]

(vgl. **Abbildung 5**). Dass dieses und kein anderes Vorgehen gewählt wird, liegt im sozialen System selbst begründet. Gewisse Kommunikation ist anschlussfähig und folgt gefestigten Regeln und Mustern, andere Kommunikation wird dadurch unwahrscheinlich.
Eine weitere Besonderheit der Luhmannschen Systemtheorie stellt die Kommunikation sozialer Systeme über deren Systemgrenzen hinweg dar. So kann, in einem vereinfachten Verständnis der Systemtheorie, die Kommunikation zwischen sozialen Systemen über Reize verstanden werden: Ein ausgesandter Reiz trifft auf die Grenze eines sozialen Systems und wird dann im Sinne einer Komplexitätsreduktion interpretiert. Konkret bedeutet dies, dass es sich dem direkten Einfluss des Senders entzieht, ob und wie der Reiz in dem den Reiz aufnehmenden System verarbeitet und interpretiert wird.

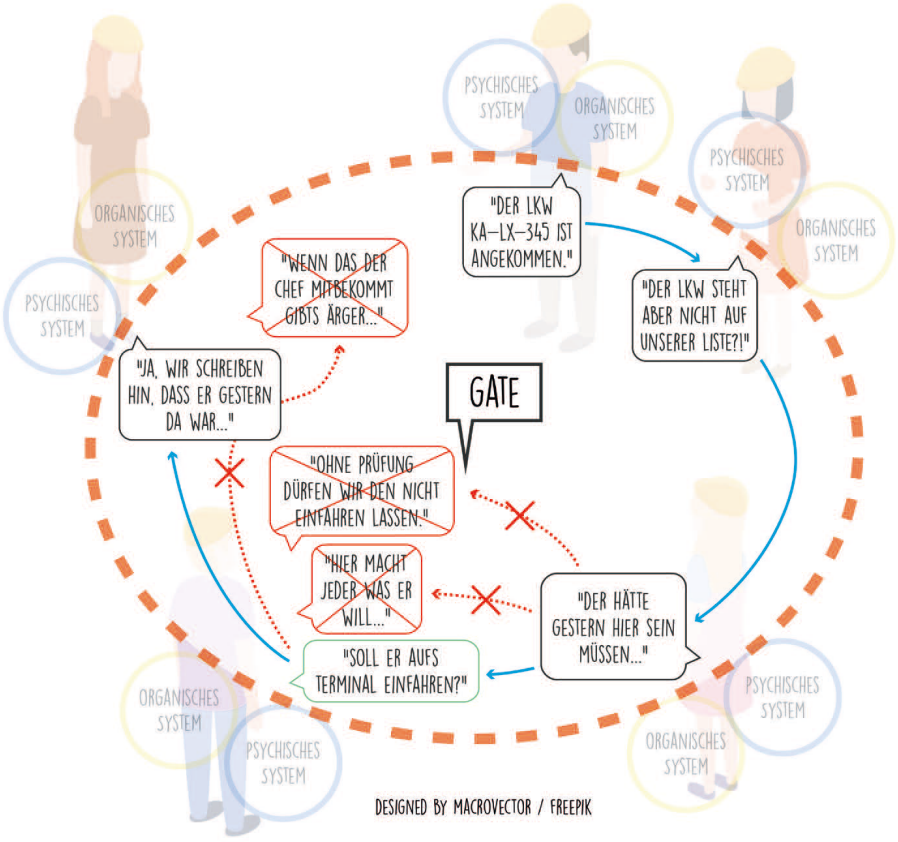

Abb. 5: Beispielhafte Kommunikation im sozialen System Gatehaus. Gezeigt ist anschlussfähige Kommunikation in Grün, und in diesem System unwahrscheinliche Kommunikationsmöglichkeiten in Rot.
Bilder: Designed by macrovector/Freepik, bearbeitet durch die Autoren

Mit diesem Wissen wird der systemischagile RE versuchen, soziale Systeme zu finden (hier kann ihm wieder die Organisationsstruktur einen Hinweis liefern). Weiter ist er aber an relevanter Kommunikation interessiert, das ist Kommunikation, die häufig stattfindet, aber im ersten Moment weder in einem Organigramm noch in formal definierten Prozessen offensichtlich wird. Zum Beispiel stellt er fest, dass folgendes Szenario immer wieder eintritt: Ein Mitarbeiter der Sales fragt für große Kunden in der Disposition an, ob noch Transportkapazitäten für kurzfristig angefragte Transportaufträge frei sind. Geschieht diese Kommunikation häufig genug, ist sie relevant, unser RE hat ein neues soziales System entdeckt. Um relevante Kommunikation aufzudecken, können Techniken aus dem systemischen Coaching oder der systemischen Therapie helfen (beispielsweise zirkuläre oder hypothetische Fragen, Skalierungs- oder Wunderfragen). Ein Mitarbeiter im Gatehaus kann Folgendes gefragt werden:

- Angenommen, Ihr Kollege aus dem Gatehaus trifft sich mit Ihrem gemeinsamen Vorgesetzten: Wie würde er Ihre Arbeitsabläufe im Gate beschreiben?

- Woran würden Sie erkennen, dass Sie einen gelungenen Arbeitstag hatten? Woran würde es Ihr Kollege aus der Disposition bemerken?

- Auf einer Skala von 1 bis 10: Wie gut können Sie Ihren Job ausführen? Wofür könnte dabei aus Ihrer Sicht die 1 und die 10 stehen? Was müsste in Ihrem Arbeitsalltag passieren, damit Sie einen Schritt weiter in Richtung der 10 gehen könnten? Was dürfte auf gar keinen Fall passieren, dass Sie einen Schritt in Richtung der 1 gehen?

- Angenommen, es würde ein Wunder geschehen und viele tägliche Herausforderungen Ihrer Arbeit würden einfach so verschwinden. Was würde sich für Sie und Ihre Kollegen im Gatehaus ändern? Was würden Sie dann konkret anders als zuvor machen?

Die gefundenen sozialen Systeme können über Reize in Beziehung stehen, und als Kontexte in einem Kommunikationssystem abgebildet werden (**vgl. Abbildung 6**).

Systemisch-konstruktivistisches Domänendesign

Sprachlich und konzeptionell weisen die Systemtheorie nach Luhmann und das DDD starke Ähnlichkeiten auf. Beide stellen die Abgrenzung von Systemen beziehungsweise Kontexten von einem Außen in den Vordergrund, Kommunikation und Sprache sind die wesentlichen Elemente, die der Konsensfindung dienen. Die (vereinfachte) Intersystemkommunikation über Reize hat ebenfalls hohe Ähnlichkeit mit der Interkontextkommunikation, wie sie über Domänenevents und einen Nachrichtenbus stattfinden kann, insbesondere wenn stark nachrichten- und serviceorientierte Architekturen gewählt werden.

In verteilten, lose gekoppelten und weitgehend autonom agierenden Teams können über asynchron entkoppelte Schnittstellen eingehende Nachrichten (aller Spezifikation und guten Wünschen zum Trotze) „anders als gewollt verstanden" oder (noch) gar nicht konsumiert werden. Dies kann analog zur Komplexitätsreduktion, wie sie an der Grenze von sozialen Systemen stattfindet, verstanden werden. Diese Analogien und Interpretationen sind

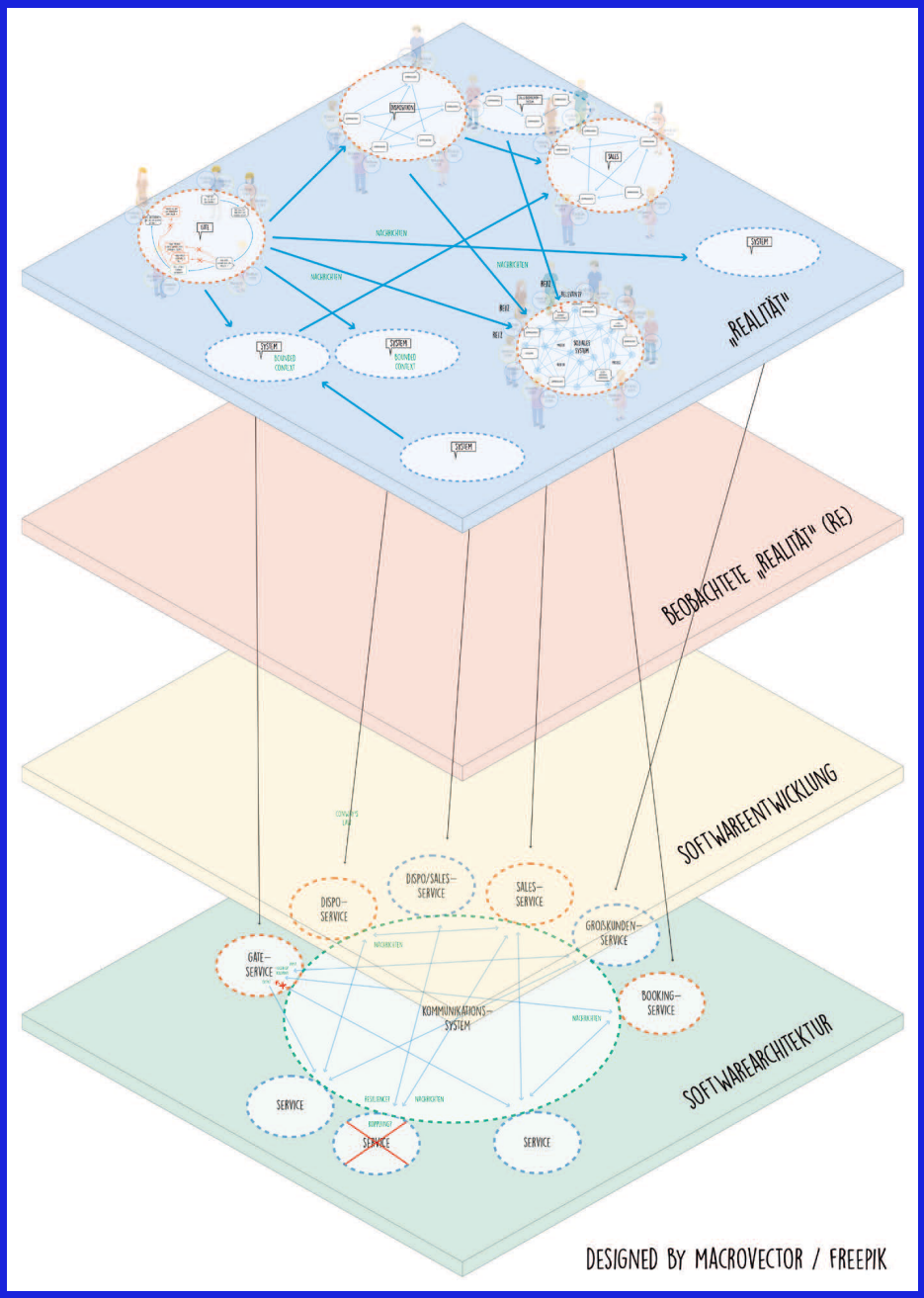

Abb. 6: Darstellung des Zusammenhangs zwischen sozialen Systemen und relevanter Kommunikation am Containerterminal mit der Abbildung der Systeme als Services in einer Softwarearchitektur.
Bilder: Designed by macrovector/Freepik, bearbeitet durch die Autoren

konform mit einer gemäß Conway's Law angenommenen isomorphen Abbildung zwischen den im Problemraum angesiedelten sozialen Systemen und den abgeschlossenen Kontexten im Lösungsraum (**vgl. auch nochmals Abbildung 6**).

Techniken des modernen DDD, die dabei unterstützen, abgeschlossene Kontexte zu finden und die Kommunikation zwischen den Aktoren zu beschreiben, behalten ihre Berechtigung. Dazu gehören unter anderem (Big Picture und Design Level) Event Storming, Domain Storytelling oder das Erstellen von Context Maps. Diese können zusammen mit den oben diskutierten Techniken, wie systemisch motivierte Fragestellungen, wertvolle Werkzeuge sein, um relevante Kommunikation (als Eventfluss) in der Fachdomäne aufzuzeigen. Dies bedeutet, die relevanten sozialen Systeme und ihre Zusammenhänge zu finden.

Iteratives Vorgehen: Ein Plädoyer

Die zuvor diskutierten Ansätze und Methoden können in einem weiteren Schritt mit einer iterativen und agilen Arbeitsweise sinnvoll kombiniert werden. Denn eine agile Vorgehensweise ist häufig die Wahl bei der Entwicklung von Software. In Accelerate [For18] ist statistisch rigoros der positive Einfluss von DevOps-Kultur und agilen Methoden nachgewiesen worden, in [Ske19] ist ebenfalls der starke Einfluss von Conway's Law auf die Zusammenstellung und Arbeitsweise von Teams beschrieben.

Die soziologische Betrachtung mittels der Systemtheorie nach Luhmann liefert eine weitere positive Betrachtungsweise: Um die Konsense kontinuierlich in Einklang zu halten, ist es notwendig, kontinuierlich zu lernen und auszurollen (continuous learning and deployment), schnell prüfen und ändern zu können (inspect and adapt) sowie falsche Entwicklungspfade schnell zu erkennen (fail fast). Anderweitig drohen die Konsense wieder auseinanderzudriften, und das Softwareprodukt verliert an Akzeptanz, oder Prozesse am Terminal gehen „am System vorbei". Beides kann fatale Folgen für die angestrebte digitale Transformation des Containerlogistikterminals haben. Zu beachten ist, dass sich Änderungen immer beidseitig auswirken. Die entstehende Software hat Einfluss auf das gesellschaftliche Miteinander der Mitarbeiter am Terminal, sie verändert die sozialen Systeme und deren Interaktion. Das wiederum ändert die Bedürfnisse an die Software.

Mit fortschreitenden Iterationen soll sich jedoch die Schnittmenge der Konsense von Entwicklern und Terminalmitarbeitern sukzessive erhöhen (**siehe Abbildung 7**): Die Software bildet anerkannte Prozesse besser ab, und Prozesse in der Organisation passen sich dem Softwareprodukt an. Iteratives Vorgehen bietet die Chance, die verschiedenen Realitätskonstruktionen zielgerichtet zu harmonisieren.

Wie geht es weiter?

Die Systemtheorie nach Luhmann erscheint uns als geeignetes Mittel, in der Problemdomäne nach geeigneten Strukturen, den sozialen Systemen, zu suchen, die sich später in der Softwarearchitektur wiederfinden. Unserer Auffassung nach ist sie konform mit und kompatibel zu zeitgemäßen Techniken, die DDD bereitstellt. Die Systemtheorie betrachtet allerdings explizit keine technischen Systeme.

Zukünftig ist die dargestellte Betrachtungsweise mit dem Begriff des soziotechnischen Systems in Verbindung zu bringen (diese werden beispielsweise in [Ske19] diskutiert). Insbesondere ist zu analysieren, was die von Conway's Law beschriebene Isomorphie bedeutet, wenn eine strukturelle Ununterscheidbarkeit von sozialen und technischen Systemen angenommen wird. Diesen Punkten wollen wir uns in Zukunft widmen, um besser zu verstehen, wie die Softwarearchitektur mit dieser soziologischen Sichtweise besser zu vereinen ist, mit dem Ziel, stabile, resiliente und evolvierbare Systeme zu bauen.

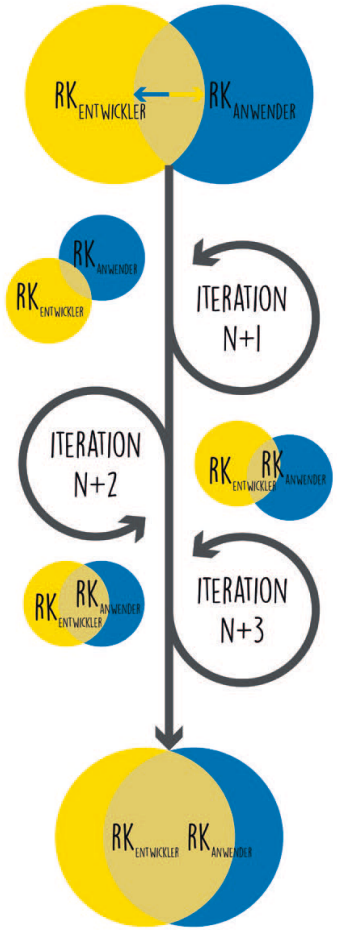

Abb. 7: Verdeutlicht anhand der Realitätskonsense (RK) von Anwendern und Softwareentwicklern, wie sich die Akzeptanz einer Software entwickeln kann. Die ersten Iterationen könnten bei den Anwendern Irritationen auslösen und die Akzeptanz negativ beeinflussen, falls zum Beispiel Arbeitsabläufe der Software anzupassen sind: blauer und gelber Kreis entfernen sich. Iteratives Arbeiten bietet die Chance, die Realitäts konstruktionen zielgerichtet zu harmonisieren, um mit fortschreitenden Iterationen die Schnittmenge beider Kreise sukzessive zu erhöhen.

Literatur & Links

[Con68] M. E. Conway, How do Committees Invent?, in: Datamation, vol. 14, no. 4, pp. 28–31
[EuroPLoP05] What do I think about Conway's Law now? Conclusions of a European Conf. on Pattern Languages of Programs 2005 Focus Group, siehe: http://www.hbs.edu/faculty/Publication%20Files/Research%20Policy%2041%20(2012)%20 1309%E2%80%93%201324_c5c2350e-013c-4065-a2f9-d95eb32177d5.pdf
[Eva03] E. J. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley Professional, 2003
[For18] N. Forsgren, J. Huble, G. Kim, Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations, IT Revolution Press, 2018
[Luh87] N. Luhmann, Soziale Systeme. Grundriß einer allgemeinen Theorie, Suhrkamp, 1987
[Luh92] N. Luhmann, Die Selbstbeschreibung der Gesellschaft und die Soziologie, 1992
[Ske19] M. Skelton, M. Pals, Team Topologies, IT Revolution Press, 2019
[Ver13] V. Vernon, Implementing Domain-Driven Design, Addison-Wesley Professional, 2013
[Wiki] siehe: https://de.wikipedia.org/wiki/Modell
[Wil82] H. Willke, Systemtheorie I: Grundlagen: Eine Einführung in die Grundprobleme der Theorie sozialer Systeme, Fischer UTB, 1982 (6. Aufl. 2000)