Requirements Engineering in IT-Ökosystemen mit Hilfe von Archetypen

Leif Singer, Eric Knauss, Kurt Schneider

Fachgebiet Software Engineering
Leibniz Universität Hannover
Welfengarten 1, 30167 Hannover
{leif.singer|eric.knauss|kurt.schneider}@inf.uni-hannover.de

Abstract: In IT-Ökosystemen ist ein breites Spektrum von Beteiligten vertreten, deren Interessen bei der Software-Entwicklung in derartigen Systemen identifiziert und gegeneinander abgewogen werden müssen. Es ist oftmals unklar, welche Eigenschaften der Beteiligten zu erwarten sind und wie sich diese auf die geplanten Requirements-Engineering-Aktivitäten auswirken werden. Diese Unsicherheit erschwert die Erhebung und Validierung von Anforderungen in solch einer Umgebung. Um diesem Problem zu begegnen, schlägt dieses Positionspapier eine Klassifizierung der möglichen Beteiligten in Archetypen vor, die die Chancen und Risiken dieser Beteiligten im Umgang mit Anforderungen durch Überzeichnung herausstellen. Mit diesem Hilfsmittel können sich Requirements Engineers ein schnelles erstes Bild der in IT-Ökosystemen zu erwartenden Herausforderungen machen. Auf Basis dieser Einschätzung können dann spezialisierte Herangehensweisen und Werkzeuge identifiziert werden, die die jeweiligen Probleme der Archetypen abschwächen und ihre Vorzüge nutzen können. Dies ermöglicht ein effizienteres und effektiveres Requirements Engineering im Rahmen der besonderen Anforderungen von IT-Ökosystemen.

1 Einführung

IT-Ökosysteme sind Systeme aus IT-getriebenen Geräten, Subsystemen und Personen, die als *Akteure* miteinander mittels der *Infrastruktur* des IT-Ökosystems interagieren. Hierbei genießt jeder Akteur des IT-Ökosystems eine beschränkte Autonomie. Er kann zeitweilig ohne direkte Weisung Entscheidungen fällen, ist aber der Kontrolle einer übergeordneten Instanz unterworfen. Als Beispiel kann man sich eine „SmartCity“ vorstellen, in der Verwaltung, Bürger, Unternehmen und Infrastruktur (wie bspw. Ampelanlagen) mit Hilfe von vernetzten Geräten interagieren und Dienste in Anspruch nehmen.

Mit der fortschreitenden Verbreitung derartiger, eng vernetzter Systeme aus Software, Geräten und Menschen muss sich auch die Art und Weise verändern, wie die Software dafür entwickelt wird. Insbesondere muss sich der Umgang mit Anforderungen ändern, denn die Randbedingungen von traditionellen Projekten und Softwareprojekten im Kontext von IT-Ökosystemen können sich grundsätzlich unterscheiden. Mit den herkömmlichen Techniken des Requirements Engineering erreicht man hier schnell die Grenzen wirksamer Partizipation.

Dieser Beitrag identifiziert typische Eigenschaften und Verhaltensweisen von an IT-Ökosystemen Beteiligten. Diese werden mit den in konventionellen Softwareprojekten für das Requirements Engineering vorausgesetzten Eigenschaften verglichen. Auf diese Weise identifizieren wir Lücken und Handlungsbedarf für den ingenieurmäßigen Umgang mit Anforderungen in IT-Ökosystemen. Der Vergleich bezieht sich auf die Aktivitäten im Requirements Engineering, die im RE-Referenzmodell auftreten [1]. Dieses unterscheidet Requirements Analysis und Management, wobei die in der Analyse genannten Aktivitäten die Folgenden sind: *Elicitation, Interpretation, Negotiation, Documentation* und *Validation & Verification*. Der Management-Teil wird durch diesen Beitrag nicht betrachtet.

Je nach der Rolle einer Person sind unterschiedliche Eigenschaften und Verhaltensweisen wichtig. Daher wird nach einem einfachen Rollenmodell differenziert, das drei charakteristische Rollen bei der Entwicklung von Software für IT-Ökosysteme unterscheidet.

In der Folge werden typische Ausprägungen der Verhaltensweisen und Eigenschaften beschrieben. Sie liegen zwischen den idealisierten und den ungünstigsten Ausprägungen einer Rolle in Bezug auf die RE-Aktivitäten. Man kann sich den Vektor dieser Eigenschaften als das Profil einer Personengruppe vorstellen. Natürlich verbietet es sich in der Praxis, die Persönlichkeitsprofile der beteiligten Stakeholder individuell zu erheben: Neben datenschutzrechtlichen Bedenken wäre schon der Aufwand unverhältnismäßig hoch.

Stattdessen beschreibt dieser Beitrag sogenannte Archetypen. Darunter verstehen wir, in loser Anlehnung an die Begriffsbildung von C. G. Jung [2], eine etwas überzeichnete Beschreibung eines Persönlichkeitstypus, der für das Verständnis gewisser Verhaltensweisen wichtig ist. Wir schlagen vor, die Abweichung häufig auftretender Archetypen zu den idealisierten Profilen für die entsprechende Rolle zu identifizieren – und durch gezielte Maßnahmen hieraus entstehende Probleme abzuschwächen.

Abschnitt 2 stellt ausführlicher die Eigenschaften eines IT-Ökosystems dar. In Abschnitt 3 wird von verwandten Arbeiten berichtet. Abschnitt 4 führt das einfache Rollenmodell für Software-Entwicklung in IT-Ökosystemen ein und spannt mit den Aktivitäten einen Raum von Eigenschaften auf, in dem aus der Warte des Requirements Engineerings idealisierte und risikobehaftete Eigenschaften ausgezeichnet werden. Dazwischen bewegen sich die Profile der Archetypen, die in Abschnitt 5 eingeführt werden. Der Beitrag schließt mit einer Zusammenfassung und einem Ausblick.

2 IT-Ökosysteme und Requirements Engineering

Ein IT-Ökosystem ist ein System, in dem verschiedene Akteure über elektronische Schnittstellen miteinander interagieren. Diese Interaktion besteht aus dem Anbieten und Nutzen von verschiedensten Diensten sowie einer gemeinsamen Infrastrukturschicht.

Ein solches IT-Ökosystem kann bspw. eine Stadt („SmartCity“) sein, in der die Verwaltung eine gewisse Infrastruktur zur Verfügung stellt, um Unternehmen die Möglichkeit zu geben, verschiedene Dienste für Bürger zur Verfügung zu stellen. Durch diesen besonderen Kontext ergeben sich für das Requirements Engineering in einem IT-Ökosystem spezielle Rahmenbedingungen. Besonders interessant sind in diesem Zusammenhang die folgenden drei Bereiche:

Infrastruktur und Rahmenbedingungen. Die Infrastrukturschicht ist in einem IT-Ökosystem Medium und Mediator für das Anbieten und Nutzen der Dienste. Insbesondere sorgt sie dafür, dass in der Interaktion der verschiedenen Akteure wichtige Rahmenbedingungen eingehalten werden. Solche Rahmenbedingungen wären bspw. die Garantie einer einheitlich sicheren Identifizierung aller Teilnehmer eines Nachrichtenaustauschs oder auch ein gemeinsames Transportprotokoll. Innerhalb der Grenzen dieser Beschränkungen agieren die Akteure autonom.

Dienste und Ziele. Ein Dienst in einem IT-Ökosystem ist eine elektronisch ansprechbare Schnittstelle zu einem möglicherweise entfernten IT-Subsystem. Das Ansprechen eines Dienstes kann etwa eine Information liefern, einen Sensor auslösen oder einen Aktuator ansteuern. Ein Beispiel ist ein per HTTP ansprechbarer Börsenticker; ein anderes ist eine Ampelsteuerung, die von der Verkehrsleitzentrale aus neu konfiguriert werden kann. Ein solcher Dienst ist nicht zu verwechseln mit Diensten in serviceorientierten Architekturen; er ist noch allgemeiner zu verstehen.

Der Anbieter eines Dienstes verfolgt durch sein Angebot bestimmte Ziele. Um dieses zu erreichen, muss er frühzeitig Beteiligte des IT-Ökosystems identifizieren, die für seine Ziele relevant sind.

Beteiligte und Rollen. Beteiligte eines IT-Ökosystems sind menschliche Akteure. Sie müssen sich ihrer Beteiligung an einem IT-Ökosystem nicht bewusst sein, um dafür relevant zu sein. Beteiligte können in einer oder mehreren der folgenden drei Rollen in einem IT-Ökosystem tätig werden: Als Vertreter des Systems, als Dienstanbieter und als Dienstnutzer. Als *System* schaffen Beteiligte Rahmenbedingungen und Infrastruktur (z.B. Stadtverwaltung einer SmartCity). Als *Anbieter* stellen sie Dienste auf Basis der Rahmenbedingungen und Infrastruktur zur Verfügung (z.B. Arztpraxis als Anbieter eines Termin-Dienstes), die dann von *Nutzern* eingesetzt werden (z.B. Arztpraxis als Nutzer eines Abrechnungsdienstes der Versicherungen).

IT-Ökosysteme müssen sich potentiell häufig ändern und anpassen – so können sich Dienste ändern, neue hinzukommen oder vorhandene entfernt werden. Aber auch die gemeinsame Infrastruktur kann neue Forderungen an die Akteure stellen, Sicherheiten wegfallen lassen oder bestehende Rahmenbedingungen ändern, um neuen Umständen gerecht zu werden. Wegen des hohen Vernetzungsgrads in diesen Systemen ist nicht leicht überschaubar, welche Auswirkungen solche Änderungen an einer Stelle des IT-Ökosystems auf andere Akteure haben. Besonders aus dem Zusammenspiel vieler Akteure entstehende emergente Effekte können bei der Einführung von Änderungen leicht übersehen werden.

Daher ist es im Rahmen von Diensterstellung, aber auch bei der Schaffung neuer Rahmenbedingungen und Infrastruktur in IT-Ökosystemen notwendig, systematisches Requirements Engineering zu betreiben. Zeitgleich ist dies aber auch eine größere Herausforderung als in der klassischen Software-Entwicklung, denn es muss nun berücksichtigt werden, dass sich die Betroffenen möglicherweise gar nicht bewusst sind, dass sie sich in einem IT-Ökosystem befinden und von dessen Dynamik beeinflusst werden.

Der im Folgenden vorgestellte Ansatz, archetypische Stakeholdergruppen zu identifizieren, um so Vorgehensweisen und Werkzeuge des Requirements Engineering besser auf die Beteiligten abstimmen zu können, soll diese den IT-Ökosystemen eigenen Probleme abschwächen.

3 Verwandte Arbeiten

Der spezielle Kontext von IT-Ökosystemen erlaubt es nicht uneingeschränkt, bestehende Ansätze des Requirements Engineering zu übernehmen. Vielmehr müssen Anleihen aus verschiedenen anderen Feldern die notwendigen Methoden bereitstellen. In diesem Abschnitt stellen wir dar, welche existierenden Ansätze ähnliche Themen berühren und warum diese nicht ausreichen, um mit der großen Bandbreite von Beteiligten in IT-Ökosystemen umzugehen.

Ansätze für das Stakeholdermanagement dienen dazu, systematisch mit den Stakeholdern eines zu erstellenden Softwaresystems umzugehen [3]. In konventionellen Projekten ist dabei das Ziel, keine relevanten Stakeholder zu vernachlässigen. In IT-Ökosystemen sind jedoch viele Beteiligte eines (Teil-)Systems auf Grund dessen emergenten Charakters nicht identifizierbar.

Spezielle Ansätze behandeln die Verwaltung von Stakeholdern und deren Wünschen in Open-Source-Projekten [4]. In Todds Ansatz geht es vor allem darum, aus einem Anwenderforum die Wünsche der Stakeholder herauszuarbeiten. Dabei stellt sich das Problem, dass ohne Moderation keine einheitliche Struktur in den Foren entsteht, so dass die gleichen Wünsche immer wieder an unterschiedlichen Stellen geäußert werden. Dieser Ansatz ist im Kontext von IT-Ökosystemen sehr interessant, da er erlaubt, Meinungen großer Mengen von Beteiligten zu handhaben. Jedoch kann nicht bei allen Beteiligten ein Interesse an dem zu erstellenden System vorausgesetzt werden, so dass die Analyse von (in einem Forum) artikulierten Wünschen große und wichtige Beteiligten-Gruppen auslässt.

Die Zielmodellierung, bspw. mit i* [5], erlaubt es, widersprüchliche Ziele von Stakeholdern explizit zu modellieren. Dies ist eine wichtige Eigenschaft im Rahmen von IT-Ökosystemen, da sowohl emergente Entwicklungen, als auch der Versuch, diese regelbasiert zu steuern, auf Befürworter, Gegner und Gleichgültige stoßen können. Wenn diese Interessensgruppen nicht zunächst identifiziert werden, wird es kaum gelingen, ein aussagekräftiges Zielmodell zu erstellen. Aus diesem Grund schlägt dieser Beitrag die Klassifikation mittels Archetypen vor.

4 Rollen, Eigenschaften und Verhaltensweisen

In Abschnitt 2 wurde ein Rollenmodell skizziert, das zwischen System, Anbieter und Nutzer unterscheidet. Dieser Abschnitt setzt diese Rollen in den Kontext der vorgeschlagenen Klassifizierung von Beteiligten in Archetypen als Basis für systematisches Requirements Engineering in IT-Ökosystemen.

Die vorgestellte Rolleneinteilung ließe sich noch weiter verfeinern, bspw. durch die Unterscheidung der Systemrolle in weitere Rollen wie Geräteentwickler, Infrastrukturentwickler, Verwaltung, etc. – für den Fokus auf die Möglichkeiten der Partizipation sind jedoch genau diese drei Rollen relevant und ausreichend.

Es ist wichtig und typisch für IT-Ökosysteme, dass die Autonomie der Akteure beschränkt ist. Dies geschieht bspw. durch Observer und Controller [6] oder andere regelnde Rückkopplungen. Diese vertreten dann die Position der Systemrolle und sind damit ebenfalls Stakeholder mit Anforderungen an die Anbieter. Außerdem vermitteln sie Anforderungen der Nutzerrolle an die Anbieter; so löst die Anforderung „ein sicheres Benutzen des Gesamtsystems“ des Nutzers an das System wiederum eine Anforderung des Systems an die Anbieter aus, die sich mit ihren Diensten dann an die entsprechenden Forderungen halten müssen. Anbieter hingegen haben Anforderungen an das System, da dies die Infrastruktur bereitstellt, auf denen die Dienste des Anbieters funktionieren müssen. Ein Überblick über diese Zusammenhänge findet sich in Abbildung 1.

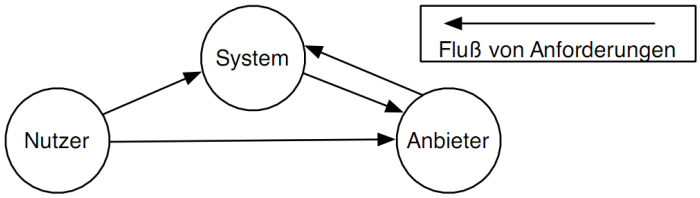

Abb. 1: Rollen und Flüsse von Anforderungen

Diese verschiedenen Flüsse von Anforderungen lassen sich nun in einer Matrix den verschiedenen Requirements-Engineering-Aktivitäten gegenüberstellen (siehe Tabelle 1).

Für jeden Eintrag in der Matrix lässt sich dann angeben, welche Eigenschaften oder Verhaltensweisen derjenigen Rolle, die Quelle der Anforderung ist, besonders förderliche oder besonders hinderliche Auswirkungen auf die jeweilige Requirements-Engineering-Aktivität haben. Hier auftauchende Risiken können dann mit geeigneten Mitteln wie Vorgehensweisen, Interviewstrategien oder Werkzeugen abgeschwächt werden. Entdeckte Vorteile hingegen können bewusst genutzt werden, um sie ebenfalls mit Hilfe entsprechender Mittel zu verstärken.

Tabelle 1 zeigt einen Ausschnitt dieser Matrix, wobei ausgewählte Beispiele für begrüßenswerte (+) bzw. risikoträchtige (–) Eigenschaften angegeben sind. Die Spaltentitel geben an, von welcher Rolle zu welcher anderen Rolle im jeweiligen Fall Anforderungen fließen sollen. So kennzeichnet „N→A“ bspw. einen Fluss von Anforderungen von der Nutzerrolle zur Rolle des Anbieters.

Tabelle. 1: Gegenüberstellung von Requirements-Engineering-Aktivitäten und Flüssen von Anforderungen mit Vor- und Nachteilen der jeweiligen Quelle des Flusses.

		N→A	**S→A**	**N→S**	**A→S**
Elicitation	**+**	Nutzer ist aufgeschlossen und nicht vorbelastet.	System kennt genaue Vorschriften.	Nutzer ist engagiert wegen Wunsches nach Sicherheit.	Anbieter ist engagiert: wirtschaftl. Interesse.
	–	Nutzer hat keine Zeit und steht neuem Dienst ablehnend geg.	System blockiert Entwicklung, ist bürokratisch und drakonisch.	Nutzer ist teilnahmslos, Meinungen von Minderheiten überrepräsentiert	Anbieter ist wegen wirtschaftl. Interesses sehr fordernd.
Interpretation	**+**	Nutzer weiß, was unaufdringlich genug ist.	Dem System sind Rahmenbedingungen klar.	Nutzer hat klares Meinungsbild.	Anbieter macht seine Ziele klar.
	–	Nutzer hat kein Interesse.	System fürchtet Rechtsunsicherheit.	Nutzer ist wankelmütig.	Anbieter verschleiert seine Ziele.
…	**+**	…	…	…	…
	–	….	…	…	…

Der Eintrag (N→A, Elicitation, –) begründet sich bspw. wie folgt: Wenn ein Nutzer einem Anbieter seine Anforderungen an einen neuen Dienst mitteilen soll, hat der Nutzer im schlimmsten Fall kein Interesse am Dienst und auch keine Zeit. Dies wäre bei einem vollkommen unbeteiligten Nutzer der Fall, der womöglich gar nicht weiß, dass er Teil eines IT-Ökosystems ist oder dessen Dienste evtl. bereits nutzt.

5 Archetypen von Stakeholdern

Bereits eine Matrix wie in Tabelle 1 ist beim Requirements Engineering hilfreich. Jede Rolle kann sich daran orientieren und das eigene Verhalten am idealen Fall ausrichten. Dies werden jedoch solche Stakeholder nicht tun, die als Nutzer wenig am Requirements Engineering interessiert sind oder sich ihrer Beteiligung an einem IT-Ökosystem nicht bewusst sind.

Für diese Art von Beteiligten muss derjenige, der die Anforderungen erhebt, Mittel und Wege finden, um Risiken zu kompensieren. Dies kann mit Werkzeugen, Techniken oder durch die Art der Befragung und Validierung geschehen. Typischerweise würden dies die Requirements Engineers tun, die im Auftrag eines Anbieters oder des Systems die Anforderungen für in das IT-Ökosystem einzubringende Änderungen erheben sollen.

Auf Basis einer solchen Übersicht über aus Warte des Requirements Engineerings idealisierte und ungünstigste Verhaltensweisen lassen sich nun Benutzerprofile erstellen, die die Position einer Benutzergruppe zwischen Ideal und größtem Risiko einordnen. Diese Profile heißen dann Archetypen und erhalten einen Titel, um eine einfach Kommunikation zu ermöglichen. Dies ähnelt den Titeln von Entwurfsmustern [7], durch die eine eigene Architektur-Sprache aufgespannt wird. Anhand der weiter unten skizzierten Beispiele für Archetypen ließe sich so zwischen Requirements Engineers bspw. ein Dialog wie der Folgende vorstellen: „Das System ist ein *Ministerium*, der Anbieter aber ein *Startup*. Das wird Probleme geben."

Noch unveröffentlichte Vorarbeiten zu IT-Ökosystemen des Fachgebiets Software Engineering der Leibniz Universität Hannover haben eine Reihe von archetypischen Verhaltensweisen zum Vorschein gebracht. Es lassen sich jedoch wahrscheinlich noch weitere Archetypen finden. Daher sind die folgenden vier Beschreibungen von Archetypen als beispielhafte Vertreter zu sehen, die die Vorgehensweise illustrieren sollen. Die beiden Archetypen *Unwissentlicher Nutzer* und *Technikfreak* sind hierbei Vertreter der Nutzerrolle des IT-Ökosystems, während das *Startup* ein Anbieter ist und das *Ministerium* die Rolle des Systems vertritt.

Archetyp „Unwissentlicher Nutzer"

Der *Unwissentliche Nutzer* ist sich nicht bewusst, dass er ein IT-Ökosystem in Anspruch nimmt oder Teil davon ist. Er sieht nur das Gesamtsystem, bzw. den Nutzen, den er davon hat, und hat aber keine Motivation, bei Anforderungsaktivitäten mitzuwirken. Mögliche Gründe hierfür sind Zeitmangel, fehlendes Interesse am System im Allgemeinen oder fehlendes Interesse oder Verständnis für IT im Speziellen.

Ein mögliches Beispiel für diesen Archetyp ist der Autofahrer, dessen Navigationsgerät sich mit der Verkehrsleitzentrale abstimmt, um möglichst optimale Routen planen zu können. Dieser Archetyp scheint am ehesten in IT-Ökosystemen zu erwarten sein, die besonders stark in den nicht IT-spezifischen Alltag wie etwa den Straßenverkehr oder Supermärkte eingebettet sind.

Archetyp „Technikfreak"

Der *Technikfreak* ist ein Nutzer, der sehr stark an Technik interessiert ist. Er ist sehr neugierig und interessiert an der IT, die ihn umgibt. Im Gespräch ist er euphorisch und idealistisch – er hat viele Ideen, kann sich dabei aber nicht auf die wirklichen Anforderungen konzentrieren. Er überlegt sich außerdem ständig, wie man seine Ideen dann umsetzen könnte. Da es in IT-Ökosystemen besonders viele Möglichkeiten auch für die private Beteiligung der Nutzer und viel autonomes Verhalten gibt, könnte dieser Archetyp hier ebenso besonders häufig vorkommen.

Ein typischer Vertreter dieses Archetyps ist bspw. ein Informatik-Forscher an einer „SmartUni" oder Informatik-affine Schüler in einer entsprechenden Schule.

Archetyp „Startup“

Ein Anbieter vom Archetyp *Startup* zeichnet sich dadurch aus, dass er ganz besonders viel Wert auf flexible Möglichkeiten in der Gestaltung seines Geschäftsmodell legt, da dies noch nicht oder nur wenig gefestigt ist. Er muss viele Strategien ausprobieren und ist daher gegenüber starren Prozessen, wie sie bspw. bzgl. Sicherheitsthemen vom System vorgeschrieben werden, eher abgeneigt.

Ein Beispiel wäre ein vor Kurzem gegründetes Unternehmen, das ein ortsbasiertes soziales Netzwerk in einer „SmartCity“ betreiben möchte, um dann mit kontextbezogener Werbung Geld zu verdienen.

Archetyp „Ministerium“

Das *Ministerium* ist eine ordnende Instanz mit hohem Regelungsbedürfnis und befindet sich in der Rolle des Systems. Es legt großen Wert auf genau vorgeschriebene Prozesse und detaillierte Schnittstellen, die alle Anbieter einhalten müssen. Neuen Anforderungen an die Infrastruktur steht es einigermaßen aufgeschlossen gegenüber, ist aber eher träge, wenn diese umgesetzt werden sollen.

Beispiele wären etwa der Bürgermeister einer als IT-Ökosystem ausgeführten Stadt oder deren Verkehrsdezernent.

Profil zum Archetyp *Unwissentlicher Nutzer*

In der folgenden Tabelle 2 wird ein Profil zum Archetyp *Unwissentlicher Nutzer* beispielhaft angedacht. Es ist angelehnt an die Matrix aus Tabelle 1. Zu jeder Requirements-Engineering-Aktivität gibt das Profil an, welche die aus Sicht des Requirements Engineerings günstigsten Eigenschaften wären (oberste Zeile, „+“), welche die entsprechend ungünstigsten Eigenschaften wären (unterste Zeile, „–“) und welche diejenigen Eigenschaften sind, die dem jeweils betrachteten Archetyp zugeschrieben werden (mittlere Zeile, „Archetyp“). Ein solches Profil bezieht sich auf eine der o.g. Interaktionen, bspw. „N→A“ (Anforderungen fließen vom Archetypen in der Rolle Nutzer zum Anbieter), etc.

In Tabelle 2 sind die günstigsten und ungünstigsten Eigenschaften ausgelassen – für dieses kurze Beispiel wird auf die Beispiele aus Tabelle 1 verwiesen.

Tabelle 2: Profil für den Archetyp „Unwissentlicher Nutzer“ in der Relation N→A.

	Elicitation	Interpretation	Negotiation	Documentation	Validation & Verification
+	…	…	…	…	…
Archetyp	Ist desinteressiert und hat keine Zeit. Weiß nichts vom IT-Ökosystem oder dem Dienst.	Möchte sich nicht wiederholen und reagiert mit Missverständnis, wenn er selbst falsch verstanden wird.	Weiß, was er nicht will, was ihm unangenehm wäre.	Hat wegen fehlenden Interesses eine geringe Aufmerksamkeitsspanne. Er versteht formale Texte und Notationen nicht.	Ist ein zufälliger einzelner Nutzer, weiß nichts von Anforderungen anderer und hat keinen Zugang dazu.
–	…	…	…	…	…

In den verschiedenen Phasen ist der befragte *Unwissentliche Nutzer* stets eine andere Person. Im Beispiel der „SmartCity“ wäre dies bspw. ein Passant, der von Requirements Engineers angesprochen wird. Er ist womöglich in Eile und hat kein Interesse an Diensten des IT-Ökosystems, da er weder einen Zugang zu IT hat noch von der Existenz des Systems weiß.

Ein möglicher Umgang mit einem solchen Nutzer wäre bspw. die Nutzung von kurzen Videos zur Validierung von bereits erfassten Anforderungen. Diese könnte man mit einem tragbaren Abspielgerät auch einem Passanten kurz zeigen, der dann zumindest sagen könnte, ob er dem gezeigten Dienst ablehnend, neutral oder positiv gegenüber steht. Dieser Ansatz wird im Fachgebiet Software Engineering der Leibniz Universität Hannover bereits exploriert.

6 Zusammenfassung

In IT-Ökosystemen müssen Requirements-Engineering-Aktivitäten oft angepasst und anders durchgeführt werden als in klassischen Software-Projekten. Dies liegt im Wesen von IT-Ökosystemen – sie sind bei Änderungen auf die Anforderungen von zum Teil unwissenden, desinteressierten und zu einer Zusammenarbeit nicht motivierten Beteiligten angewiesen.

Dieser Beitrag versucht, einen möglichen Weg aufzuzeigen, der diesen spezifischen Gegebenheiten gerecht wird. Hierzu wurden im Raum aus Aktivitäten und den an IT-Ökosystemen beteiligten Rollen typische Muster identifiziert, indem über die Eigenschaften der verschiedenen Rollen in ihren jeweils extremsten Ausprägungen nachgedacht wurde.

Zwischen diesen Extremen wurden typische Ausprägungen in einem Eigenschaftsprofil zu Archetypen zusammengefasst. Einige dieser Archetypen kommen, durch die speziellen Gegebenheiten in IT-Ökosystemen hervorgebracht oder begünstigt, häufig vor. Um Differenzen besser sichtbar zu machen, wurden die Eigenschaften der Archetypen überzeichnet und pauschalisiert. Hieraus leitet sich auch die Namensgebung „Archetyp“ ab.

Es ist noch zu evaluieren, wie die besonderen Risiken häufig vorkommender Archetypen abgeschwächt werden könnten. Dies wurde vom Fachgebiet Software Engineering der Leibniz Universität Hannover für den Archetyp *Unwissentlicher Nutzer* anhand einer unveröffentlichten Fallstudie zur videobasierten Validierung von Anforderungen bereits begonnen. Für andere Archetypen sind ähnliche Explorationen empfehlenswert, um geeignete Mittel des Requirements Engineering für die jeweiligen Chancen und Risiken eines Archetyps erfahrungsbasiert auswählen zu können. Bspw. könnte erprobt werden, ob die Begeisterung des *Technikfreaks* durch technikorientierte Crowdsourcing-Initiativen, die aus vielen Ideen die besten auswählen würden, kanalisiert werden kann.

Dieser Beitrag hat eine Vorgehensweise herausgearbeitet, die von der Charakterisierung günstiger und ungünstiger Eigenschaften über die Beschreibung von Archetypen bis zu gezielten Verbesserungen führt. Die konkreten Eigenschaften, Rollen und Archetypen sind mit der Hoffnung auf Diskussion und Austausch zunächst exemplarisch angegeben. Es ist davon auszugehen, dass sich das Verfahren so an die spezifischen Situationen in anderen IT-Ökosystemen anpassen lässt und der beschriebene Kern in den meisten IT-Ökosystemen aber so bereits einen gangbaren Ansatz liefert.

Literaturreferenzen

[1] Schneider, K.: Anforderungen und Entwurf. Vorlesung, Leibniz Universität Hannover, Hannover, 2009.

[2] Jung, C. G.: Archetypen. Deutscher Taschenbuch Verlag, München, Januar 2001.

[3] Rupp, C.: Requirements-Engineering und –Management. Hanser, 4. Auflage, 2007

[4] Todd, K.; Cleland-Huang, J.: A Dynamic Approach for Managing Stakeholder Credentials in Lightweight, Collaborative, Requirements Elicitation. First International Workshop on Managing Requirements Knowledge MaRK'08. Barcelona, 2008.

[5] Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97). Washington D.C., 1997.

[6] Cakar, E; Hähner, J.; Müller-Schloer, C.: Investigation of Generic Observer/Controller Architectures in a Traffic Scenario. GI Jahrestagung, 2008.

[7] Gamma, E.; Helm, R.; Johnson, R. E.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley Longman, Amsterdam, 1995.