4 Methoden und Werkzeuge

4.1 Use-Case-basiertes Vorgehen zur Auswahl von Social Software

Sebastian Behrendt, Matheus Hauder, Marin Zec, Sascha Roth, Alexander Richter

4.1.1 Ziel und Motivation

Wie im Beitrag von Ehms/Richter erläutert, zeichnet sich Social Software insbesondere durch ihre Nutzungsoffenheit aus: Die Software selbst lässt die Art und Weise der späteren Nutzung größtenteils offen und ist nicht an ein typisches Nutzungsszenario gebunden. Hier unterscheidet sich Social Software auch bei der Softwareauswahl von traditionellen betrieblichen Anwendungssystemen (wie z. B. im Bereich Enterprise Resource Planning (ERP)), denen bereits in der Entwicklungsphase klare Strukturen, Prozesse und vorher vorgegebene bzw. definierte Nutzungsszenarien zugrunde liegen. Eine Bestandserfassungsmaske in einem ERP-System bspw. dient eben nur genau dem Zweck, Waren zu erfassen. In der Regel lässt sich sogar eine Funktion (z. B. manuelle Wareneingangskontrolle) genau einem konkreten Geschäftsprozess (z. B. Einkauf) zuordnen. Bei Social Software gibt es dagegen in der Regel verschiedene Funktionen zur Unterstützung derselben Arbeitspraktik und dieselbe Funktion kann meist zur Unterstützung unterschiedlicher Arbeitspraktiken eingesetzt werden. Das Freitextfeld in einem Microblog bspw. kann u. a. dazu verwendet werden, Probleme zu lösen, Ideen zu sammeln oder transparent zu kommunizieren. Aus diesem Grund stellt die Auswahl von Corporate Social Software die beteiligten Personen vor eine besondere Herausforderung, denn Nutzungsoffenheit führt dazu, dass klassische Software-Auswahl und -Einführungsprozesse nicht funktionieren. Insbesondere trifft dies auf die Anforderungsanalyse und Plattformauswahl zu. Grund dafür ist, dass man hier nicht funktionszentriert vorgehen kann, denn ein und derselbe Zweck sind meist mit komplett unterschiedlichen Funktionen erreichbar. Aus diesem Grund wurde ein Vorgehen auf höherer Ebene – siehe aperto-Ebenenemodell – auf der Basis von Use-Cases entwickelt, welches im vorliegenden Beitrag näher erläutert wird.

4.1.2 Anforderungen an Software im Unternehmensumfeld

Grundlage jeder Werkzeugauswahl ist die Ableitung von Unternehmensanforderungen. Um diese Anforderungen an Softwareprodukte zu strukturieren, existieren verschiedene Modelle. Dazu zählen etwa FURPS+ (Grady et al. 1987) und die internationale Norm ISO/IEC 25000, die wir im Folgenden kurz erläutern. *FURPS* steht als Akronym für die vermeintlich wichtigsten Kriterien für Softwarequalität. Das „+“ wurde im Laufe der Zeit hinzugefügt, um weitere Anforderungsaspekte, bspw. in Bezug auf Design, Implementierung, Schnittstellen oder physikalische Bedingungen, zu berücksichtigen.

- **Funktionalität** (**F**unctionality): Die Fähigkeit eines Softwareproduktes, eines Moduls oder einer Komponente, eine bestimmte Aufgabe, Funktion oder Gruppe von Funktionen zu erfüllen.
- **Benutzbarkeit** (**U**sability): Das Ausmaß, in dem ein Softwareprodukt durch bestimmte Benutzer in gegebenen Anwendungskontexten genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen (ISO 9241-11).
- **Zuverlässigkeit** (**R**eliability): Die Zuverlässigkeit eines Softwareprodukts gibt an, wie verlässlich durch das Produkt abgedeckte Funktionalität in einem bestimmten Zeitintervall erfüllt wird.
- **Leistung/Effizienz** (**P**erformance): Das Verhältnis zwischen der Leistung des Softwareprodukts und den eingesetzten Mitteln unter festgelegten Bedingungen.
- **Änderbarkeit** (**S**upportability): Der Aufwand, der zur Umsetzung von vorgegebenen Änderungen (Korrekturen, Umgebung, Funktionalität) notwendig ist.

Software Product Quality Requirements and Evaluation (SQuaRE) ist die Norm ISO/IEC 25000 bzw. 25010 und ersetzt seit dem Jahr 2005 die Norm ISO/IEC 9126 (ISO 25010). *SQuaRE* betrachtet sowohl Produktqualität als auch Qualitätseigenschaften im praktischen Nutzungskontext (quality in use).[99]

In Anlehnung an diese beiden Modelle (FURPS+ und SQuaRE) konzentriert man sich im Rahmen des hier vorgestellten Vorgehens nur auf die zwei grundlegenden Kategorien von Anforderungen: funktionale und nichtfunktionale Anforderungen (Pohl 2010).

- **Funktionale Anforderungen (FA):** Als funktional bezeichnet man Anforderungen, die festlegen, was eine Software tun soll. Sie beschreiben die Anforderung an Funktionen, Daten und das fachliche Verhalten der Software. Beispiele für funktionale Anforderungen an Social Software sind: „Das System bietet dem Nutzer die Möglichkeit, beliebige Inhalte im PDF-Format herunterzuladen.“ und „Das System muss Systemartefakte versionieren und dem Nutzer den Wechsel zwischen verschiedenen Versionen eines Artefakts ermöglichen.“
- **Nichtfunktionale Anforderungen (NFA):** Unter nichtfunktionalen Anforderungen versteht man Anforderungen an bestimmte Qualitätseigenschaften eines Softwaresystems. Beispiele für nichtfunktionale Anforderungen an Social Software sind: „Das System muss in Java programmiert sein.“ und „Die Farbgestaltung der Oberfläche müssen an das Corporate Design anpassbar sein.“

4.1.3 Use-Case-basiertes Vorgehen

Ein funktionsorientiertes Vorgehen bei der Auswahl von Social Software ist jedoch aufgrund der Nutzungsoffenheit nicht angemessen. Um dennoch funktionale und nichtfunktionale Anforderungen ableiten zu können, wird ein Use-Case-basiertes Vorgehen bei der Auswahl einer Enterprise Social Software vorgeschlagen. Dieses Vorgehen besteht aus vier Schritten:

---

[99] Zu den Merkmalen der Produktqualität zählen Functional Suitability, Performance Efficiency, Compatibility, Usability, Reliability, Security, Maintainability sowie Portability. Anwendungsbezogene Qualitätsmerkmale sind Effectiveness, Efficiency, Satistfaction, Freedom from Risk und Context Coverage.

1. Identifikation von Use Cases, 2. Ableiten von Anforderungen, 3. Vorauswahl und 4. Evaluation der Plattformen. Dieses Vorgehen wurde so oder in ähnlicher Form in den vergangenen beiden Jahren in verschiedenen Praxisprojekten angewendet (vgl. Richter et al. 2012).

1. Identifikation von Use Cases

Zuerst werden Use Cases identifiziert, die mit der zukünftigen Plattform umgesetzt werden sollen. Die Ermittlung der Use Cases kann durch das Projektteam direkt geschehen, durch Vorgaben aus dem Management oder auch durch Interviews mit den zukünftigen Nutzern. Letztgenanntes Vorgehen bringt in der Regel sehr hilfreiche und oftmals für alle Beteiligten überraschende Ergebnisse mit sich.

Das Konzept **Use Case** ist vor allem in der Softwareentwicklung weitverbreitet. Sie werden verwendet, um die Menge von Anwendungsszenarien des zu entwickelnden technischen Systems zu beschreiben.[100]
Wenn im Folgenden von Use Cases gesprochen wird, dann ist damit Folgendes gemeint: Innerhalb eines Geschäftsprozesses finden mehrere kollaborative Prozesse statt. Hier handelt es sich um die (IT-gestützte) Interaktion zwischen mehreren Personen im Rahmen einer gemeinsamen Aktivität mit dem gemeinsamen Ziel, den Geschäftsprozess voranzutreiben. Ein solcher kollaborativer Prozess kann auf unterschiedliche Art und Weise umgesetzt, also durch verschiedene Szenarien abgebildet werden.
In Abgrenzung zum Begriff *Use Case* in der Softwareentwicklung wird im Rahmen dieses Beitrags der Begriff *Collaborative Use Case* (*CUC*) verwendet.
Ein *Collaborative Use Case* ist gekennzeichnet durch:
Ziele: Welche Ziele sollen erreicht werden? Was ist das angestrebte Ergebnis?
Akteure: Welche Personen(-gruppen) spielen dabei eine Rolle und was sind deren Eigenschaften?
Ablauf: Welche Schritte sind nötig, um die angestrebten Ziele zu erreichen?

Zur Identifikation von Collaborative Use Cases werden folgendee Schritte vorgeschlagen:

[100] Cockburn (2001) unterscheidet hierbei zwischen Use Case und Szenario. Ein Szenario ist eine Sequenz von Interaktionen, die unter bestimmten Bedingungen ablaufen, um das Ziel des Hauptakteurs zu erreichen und ein bestimmtes Resultat bezüglich des Ziels zu erlangen. Ein Anwendungsfall (engl. use case) bündelt alle möglichen Szenarien, die eintreten können, wenn ein Akteur versucht, mithilfe des betrachteten Systems ein bestimmtes fachliches Ziel zu erreichen.

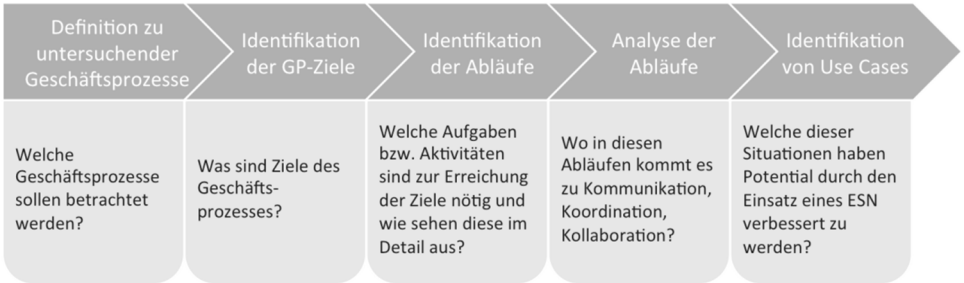

Abbildung 1: Vorgehen zur Identifikation von Collaborative Use Cases

Ausarbeiten der Use Cases: Sobald die potenziellen Use Cases identifiziert sind, müssen sie detailliert ausdefiniert werden. Dazu erfolgt eine genaue Beschreibung der Ziele, des Ablaufes und der beteiligten Akteure. Für ein besseres Verständnis kann der Ablauf auch z. B. in Form eines Ablaufdiagramms visualisiert werden. Die detaillierte Aufschlüsselung der einzelnen Arbeitsschritte bildet die Grundlage zur Ableitung von konkreten Anforderungen. In der Praxis zeigt sich hier sehr deutlich, wie unterschiedlich das Verständnis der verschiedenen Mitarbeiter über vermeintlich bekannte und definierte Prozesse ist. Die Entwicklung der Use Cases und vor allem die grafische Darstellung des Ablaufes fördert ein gemeinsames Verständnis sowohl von den angestrebten als auch den bestehenden Prozessen.

2. Ableiten von Anforderungen

Sobald die Use Cases identifiziert und ausgearbeitet sind, können daraus die Anforderungen an das zukünftige System abgeleitet werden. Für die funktionalen Anforderungen helfen dabei ***Collaborative Usage Patterns*** und ***Critical Incidents***, die im Folgenden erläutert werden.

Collaborative Usage Pattern: Bei der Ausarbeitung der Use Cases wurden die einzelnen Aktionen innerhalb des Use Cases beschrieben. Diese werden nun mithilfe der CUP-Matrix aus dem aperto-Rahmenwerk analysiert (Richter et al. 2012). Dabei stellt sich die Frage, welche CUPs sich hinter einer bestimmten Aktion verbergen. Diese ergeben dann die Anforderungen. Es kann vorkommen, dass eine Anforderung in verschiedenen Ausprägungen benötigt wird. Diese werden dann entsprechend definiert.

Das folgende Beispiel veranschaulicht diese Ableitungsmethode:

Exemplarischer Ablauf des CUC	**CUPs / Anforderung**	**Ausprägung**
Mitarbeiter stellt eine Frage an (alle anderen) Mitarbeiter	Teilen einer Nachricht	**Min:** Nachricht an alle Personen **Med:** Nachricht an ausgewählte Personen **Max:** Nachricht an ausgewählte Gruppen
Beantworten der Fragen durch andere Mitarbeiter	Teilen einer Nachricht Hinweisen auf ein Dokument Teilen eines Dokumentes	**Min:** Dokument-Link manuell kopieren **Med:** Dokument-Link In-Place suchen **Max:** Dokument-Link und Vorschau anzeigen
Verfasser der Frage benennt die „hilfreichste Antwort“ oder die „richtige Antwort“	Bewerten einer Nachricht Kennzeichnen einer Nachricht	**Min:** Taggen nur durch Verfasser **Med:** privates Taggen durch alle **Max:** offenes Taggen durch alle
häufige Fragen ggf. an prominenter Stelle hervorheben	Sammeln von Nachrichten Kennzeichnen von Nachrichten	**Min:** Sammeln in privater Liste **Med:** Sammeln in gemeinsamer Liste **Max:** Sammeln in mehreren Listen

Tabelle 1: Ableitung von Anforderungen aus einem Use Case

Die so ermittelten Anforderungen scheinen im Vergleich zu Anforderungen aus dem Lastenheft z. B. eines ERP-Systems noch etwas unkonkret zu sein. Dies ist jedoch beabsichtigt und darin liegt gerade der Vorteil dieses Vorgehens. So ist bspw. beim „Teilen einer Nachricht“ nicht explizit definiert, wie das geschehen soll. Schließlich kann dies in verschiedenen Plattformen unterschiedlich umgesetzt sein. So ermöglicht eine Plattform das Stellen einer Frage über das Status-Update-Feld. Dagegen kann eine andere Plattform einen eigens definierten Inhaltstyp namens „Frage“ haben. In beiden Fällen ist die Anforderung erfüllt und dieser Schritt des Use Cases ließe sich daher realisieren. Diese verschiedenen Arten der Umsetzung sind im Sinne der Ausprägung (siehe Spalte 3 in Tabelle 1) zu berücksichtigen. An dieser Stelle können auch Plug-ins eine wichtige Rolle spielen, da sie spezifische Funktionen nachrüsten.

Critical Incidents: Unter *Critical Incidents* verstehen wir Situationen, die von der Norm bzw. vom Standardablauf abweichen. Diese helfen dabei, Anforderungen zu formulieren, die eventuell bisher nicht ermittelt wurden. Dabei werden alle zuvor definierten Aktionen im Use Case hinsichtlich dieser Critical Incidents untersucht.

Um diese Critical Incidents zu „generieren“, sollten vor allem die nachfolgenden Bereiche betrachtet werden:

Thema	Beispiel	Anforderung
Rollen und Rechte	im Vorstandblog dürfen nur bestimmte Personen schreiben	Definition unterschiedlicher Rollen möglich
Fristen, Zeiträume, Zeitpunkte	nach zwei Wochen wird die Umfrage geschlossen	automatisches Deaktivieren der Abstimmungsfunktion
Abhängigkeiten	nur ein freigegebenes Dokument darf in bestimmten Gruppen veröffentlicht werden	verschiedene Dokumentenstatus möglich
Automatismen/Prozesse	eine Gruppe wird nach gewisser Inaktivität geschlossen	Gruppen schließen/deaktivieren
Integration	Dokumente aus dem bestehenden DMS sollen zugänglich sein	Integration eines DMS

Tabelle 2: Themenbereiche zur Entwicklung von Critical Incidents

Nachdem alle Use Cases auf diese Art analysiert wurden, sind einige Anforderungen womöglich doppelt vorhanden, da sie in mehreren Use Cases zum Tragen kommen. Eine Konsolidierung aller Anforderungen ist jedoch nicht vorgesehen. Dies hat Vorteile bei der Evaluation, auf die später noch genauer eingegangen wird.

Nichtfunktionale Anforderungen: Durch die Ableitung von CUPs und die Betrachtung von Critical Incidents lassen sich hauptsächlich funktionale Anforderungen ermitteln. Unter Umständen haben sich so auch schon einige NFA ergeben. Diese lassen sich jedoch nicht umfassend direkt aus den Use Cases ermitteln, da sie größtenteils von übergeordneter Natur sind.

In Anlehnung an die oben bereits genannte ISO-Norm und aus der Erfahrung aus mehreren Praxisprojekten heraus werden folgende zu untersuchenden Aspekte vorgeschlagen:

Bereich	Erklärung
Integration	Welche Möglichkeiten zur Integration mit anderen (bereits bestehenden) Systemen werden benötigt?
Erweiterbarkeit	Welche Möglichkeiten bestehen, die Anwendung durch Add-Ons oder Nutzung einer API an eigene Bedürfnisse anzupassen?
Rechte und Rollen	Welche Sonderrechte und Rollen werden benötigt? Wie soll das System an bestehende Rechteverwaltung (z. B. LDAP) angeschlossen werden?
Wartbarkeit	Welche Bestimmungen zwecks „Aufräumen“, Archivieren und Löschen müssen eingehalten werden?
Benutzbarkeit	Welche Anforderungen bestehen hinsichtlich des mobilen Zugangs, der Barrierefreiheit oder des Supports?
System	Welche technischen Voraussetzungen müssen berücksichtigt werden? (bestehende Server- und Technologielandschaft)
Kosten	Welcher Kostenrahmen ist zu beachten?

Tabelle 3: Überblick nichtfunktionaler Anforderungen

Die nichtfunktionalen Anforderungen werden im Anschluss zusammengefasst. Mögliche doppelte Anforderungen werden gestrichen.

Das Ergebnis der Anforderungsermittlung ist daher eine Liste mit nach Use Cases gruppierten funktionalen Anforderungen und eine weitere Liste mit nichtfunktionalen Anforderungen. Diese gilt es nun zu evaluieren.

3. Vorauswahl

Nachdem die Anforderungen erhoben wurden, sollen die Werkzeuge dahingehend evaluiert werden. Es können jedoch nicht alle prinzipiell verfügbaren Werkzeuge untersucht werden. Zuvor muss eine geeignete Vorauswahl stattfinden. Um eine umfassende Übersicht über Anbieter von Enterprise Social Software bzw. deren Produkte zu erstellen, bieten sich verschiedene Quellen an. Neben einer umfassenden Webrecherche können Analystenberichte wie z. B. der Gartner Magic Quadrant for Social Software in *the Workplace* (Mann/Gotta 2013) oder Fallstudienplattformen im Internet wie z. B. e20cases.org hilfreich sein. Bei der Recherche ist zu beachten, dass Enterprise-Social-Software-Plattformen spezifische Funktionalitäten oft nur durch entsprechende Plug-ins anbieten. Daher ist es zusätzlich zur Plattformrecherche durchaus ratsam, auch verfügbare, relevante Plug-ins und Erweiterungen zu ermitteln. Auf Basis dieser Recherche entsteht schließlich eine sogenannte Longlist.

Für die zielorientierte Kürzung der Longlist werden bestimmte NFA als Ausschlusskriterien herangezogen. Diese sind stark von den Präferenzen und Bedingungen des Unternehmens abhängig und können daher variieren. Die Erfahrung hat jedoch gezeigt, dass vor allem folgende NFA helfen können, eine sinnvolle Vorauswahl zu treffen und die Longlist somit einzuschränken.

Anforderung	Beschreibung
Hosting	Wo soll die Anwendung gehostet werden? Nur intern oder ist auch externes Hosting möglich? Und wenn ja, in welchen Ländern darf extern gehostet werden (Datenschutzgesetze)?
Reife	Wird ausdrücklich ein möglichst ausgereiftes Produkt angestrebt oder kann es auch ein Produkt sein, welches noch rapide weiterentwickelt wird (und man möchte sich dort eventuell einbringen)?
Technologie	Sind bestimmte Technologien (Webserver, Datenbanken o. ä.) durch die Unternehmens-IT vorgegeben?
Mehrsprachigkeit	Ist Mehrsprachigkeit notwendig? Wenn ja, in welchem Umfang?
Support	Ist ein In-House oder Vor-Ort-Support gewünscht?
mobile Nutzung	Ist der mobile Zugriff gewünscht und welche Rahmenbedingungen stellt hier die Unternehmens-IT?

Tabelle 4: Kriterien zur Vorauswahl

Die Ausprägungen der verschiedenen Anforderungen sind abhängig vom jeweiligen Unternehmen. Hierbei gibt es kein „gut“ oder „schlecht“. Es handelt sich vielmehr um grundlegende Bedingungen, die aus verschiedenen Gründen bestehen und erfüllt werden müssen. Beispielsweise können Shortlists zweier Unternehmen voneinander abweichen, wenn verschiedene Standardtechnologien zum Einsatz kommen oder den Projektverantwortlichen vonseiten der IT-Strategie unterschiedliche Vorgaben gemacht wurden. Es kann auch sein, dass in einem Unternehmen von Anfang an großer Wert auf eine möglichst einfache Integration in beste-

hende Systeme und Prozesse gelegt wird, während ein anderes Unternehmen vor der Einführung ein eigenständiges separates Projekt aufsetzt, in welchem erst einmal der Umgang mit solchen Plattformen erlernt werden soll.

Das Ergebnis der Filterung anhand von NFA-Ausschlusskriterien ist die sogenannte Shortlist, die eine Auswahl an Plattformen enthält, welche realistisch im Rahmen des Auswahlprozesses evaluiert werden können.

4. Evaluation der Plattformen

Evaluation der funktionalen Anforderungen: Für die Evaluation der funktionalen Anforderungen werden die zuvor identifizierten und ausgearbeiteten Use Cases in den Plattformen von der Shortlist „durchgespielt". Dabei wird geprüft, ob und in welchem Umfang sich die identifizierten CUPs abbilden lassen. Dies wird in einem dafür geeigneten Werkzeug festgehalten.

Folgender Aufbau wird vorgeschlagen:

1	**2**	**3**	**4**	**5**	**6**
ID	**Use Case**	**Anforderung**	**Vorhanden**	**Erfüllungsgrad**	**Punkte**
1	Frag den Kollegen	Nachricht mitteilen (Frage stellen)	**ja** / nein	**min**/med/max	3
N	…	…	…	…	…
Zusammenfassung					
	Frag den Kollegen		20 von 34		150

Tabelle 5: Beispiel eines Evaluationstemplates

Spalte 1 enthält eine eindeutige ID zur einfachen Referenzierung. Spalte 2 enthält den Namen des Use Cases, aus dem die Anforderung stammt. Spalte 3 enthält die Anforderung in Form des CUP. Für ein besseres Verständnis kann die ursprüngliche Aktion zusätzlich noch erläutert werden. Spalte 4 enthält die Bewertung, ob die Anforderung erfüllt wurde. Hier kann man z. B. in Excel mit einer vorgegebenen [Ja/ Nein-]Auswahlliste arbeiten. In Spalte 5 wird angegeben, in welchem Umfang diese Anforderung erfüllt ist. Spalte 6 enthält einen von Spalte 5 abhängigen Punktewert. Er bietet die Grundlage für die Zusammenfassung und Auswertung (z. B. am Ende der Liste).

Wie oben schon angesprochen, kann es vorkommen, dass manche Anforderungen wiederholt auftauchen. Diese sollten aber dennoch mehrfach, also im Rahmen des jeweiligen Use Cases, geprüft werden. Sonst bestünde die Gefahr, dass eine konkrete Funktion zwar existiert und die Anforderung somit generell erfüllt wäre, diese Funktion jedoch im Ablauf eines anderen Use Cases, der sie ebenso benötigt, nicht zu erreichen ist. Damit wäre der Use Case nicht abbildbar, obwohl die Funktion vorhanden ist. Des Weiteren kann es sein, dass in verschiedenen Use Cases ähnliche Aktionen bestehen, diese jedoch aus praktischen Gründen durch verschiedene Funktionen umgesetzt werden (siehe n:n-Beziehungen zwischen Funktionen und CUP im aperto-Rahmenwerk).

Evaluation der nichtfunktionalen Anforderungen: Die Evaluation der NFA erfolgt ähnlich der funktionalen Anforderungen. Auch hier kann es helfen, die Anforderungen strukturiert abzuprüfen (z. B. in Tabellenform). Die Erfüllung einiger NFA kann jedoch nicht durch eine klare Ja/Nein-Entscheidung beurteilt werden (z. B. die Frage nach der Servertechnologie oder den

Lizenzkosten). Hier ist vielmehr eine qualitative Beurteilung erforderlich, etwa in Form einer textuellen Beschreibung.

Auswertung: Die Auswertung erfolgt im ersten Schritt quantitativ. Das bedeutet, die erreichten Punktzahlen werden entsprechend aufsummiert und bei Bedarf z. B. als Diagramm visualisiert. Somit kann die Eignung der Plattformen für die definierten Use Cases besser verglichen werden. Wie oben beschrieben, ist ein Teil der Anforderungen nur qualitativ zu beurteilen. Eine quantitative Auswertung ist in diesem Fall nicht möglich. Auch wenn die Zahlen schon eine grobe Tendenz aufzeigen, sollten sie nicht als alleiniges Entscheidungskriterium herangezogen werden. Vor allem die qualitativen NFA und das „Gefühl“, das sich während und infolge der Testphase eingestellt hat, spielen eine wichtige Rolle. Die endgültige Entscheidung für oder gegen eine Plattform sollte daher schlussendlich auf qualitativer Ebene erfolgen. Die Zahlen dienen eher dazu, ein besseres Verständnis zu bekommen und die Entscheidung besser zu begründen.

In der Praxis überraschend war, dass obwohl teilweise dieselben Use Cases untersucht wurden, die Evaluationsergebnisse dieser am Ende unterschiedlich waren. Das lag unter anderem daran, dass der Umsetzungsgrad der einzelnen Funktionen in den Unternehmen unterschiedlich bewertet wurde. Was für das eine Unternehmen als Maximum gilt, kann für ein anderes Unternehmen nur Minimum sein (oder umgekehrt). Das liegt vor allem an den Prozessen, die Use Cases zugrunde liegen, und den Vorstellungen davon, wie diese in Zukunft ablaufen sollen. So wurden, obwohl die Longlist in vielen Praxisprojekten übereinstimmte und auch die Shortlist sowie die betrachteten Use Cases sich sehr ähnelten, letztendlich verschiedene Plattformen ausgewählt.

4.1.4 Anwendung, Kritik, Ausblick

Das geschilderte Vorgehen liefert eine Heuristik, um mit der Herausforderung der Nutzungsoffenheit von Social Software während des Auswahlprozesses umzugehen. Es wurde so oder in ähnlicher Form in verschiedenen Praxisprojekten angewendet. Wie sich aktuell dabei zeigt, unterliegen die NFA, die bei der Vorauswahl (*Shortlisting*) betrachtet werden, einem stetigen Wandel. So ist momentan eine hohe „Reife“ noch ein Kriterium, das viele Plattformen von vornherein ausschließt, da bisher nur wenige Anbieter „ausgereifte“ Produkte anbieten. Dieses Kriterium wird jedoch zunehmend weniger relevant.

Des Weiteren wurde deutlich, dass das Ergebnis einer Evaluation immer auch von den Personen abhängt, die sie durchführen (Tester). Ist ein Tester geübter im Umgang mit solchen Plattformen, so löst er Use Cases eventuell anders oder weiß, die Funktionen der Plattform und eventuell verfügbarer Plug-ins besser zu nutzen. Dies wirkt sich auf die Bewertung aus. Obwohl am Ende ein quantitatives Ergebnis ermittelt wird, fließen dennoch viele subjektive und qualitative Faktoren ein. Die Anwendung in der Praxis hat daher gezeigt, dass trotz des möglichst quantitativen Vorgehens und Resultats die endgültige Entscheidung für oder gegen eine Plattform letzten Endes einen stark qualitativen Charakter hat.

Ein genereller „Marktüberblick“ von bestehenden Plattformen lässt sich durch dieses Vorgehen somit nur auf Basis von Musteranforderungen erstellen. Vor allem die Einschätzung der NFA ist sehr vom jeweiligen Unternehmen und dessen Rahmenbedingungen abhängig. Auch die Critical Incidents, die in gewisser Weise die Details der verschiedenen Unternehmenspro-

zesse und Regularien widerspiegeln, können nicht verallgemeinert werden. Der sich daran anschließende Umsetzungsgrad der funktionalen Anforderungen (min/med/max) ist stark vom Unternehmenskontext abhängig. Das Ergebnis einer solchen Evaluation ist daher nur für das entsprechende Unternehmen aussagekräftig.