HMD Praxis der Wirtschaftsinformatik (2024) 61:1180–1212
https://doi.org/10.1365/s40702-024-01109-9

SCHWERPUNKT

Ein Werkzeug zur Analyse von Komplexität von Low-Code und No-Code Add-ons: Ein Design Science Ansatz

Adrian Abendroth · Benedict Bender

Eingegangen: 15. Februar 2024 / Angenommen: 14. August 2024 / Online publiziert: 30. August 2024

Zusammenfassung In der schnelllebigen Geschäftswelt von heute ist es für Unternehmen entscheidend, sich rasch an veränderte Marktbedingungen anzupassen, um ihre Wettbewerbsfähigkeit zu sichern. Flexibilität und maßgeschneiderte Prozesse sind dabei zentrale Erfolgsfaktoren, die oft über die Möglichkeiten standardisierter Anwendungssysteme hinausgehen. Low-Code-Plattformen (LCP) wie Mendix und Outsystems sowie No-Code-Plattformen (NCP) wie Bubble bieten vielversprechende Ansätze zur Realisierung individueller Anpassungen durch visuelle und deklarative Techniken, die herkömmliche Programmierung ersetzen.

Vor der Adoption einer solchen Plattform müssen Unternehmen beurteilen, wie gut eine spezifische LCP/NCP ihre individuellen Anforderungen erfüllen kann. Der mögliche Anforderungserfüllungsgrad hängt nicht nur von den Grundfunktionen einer Platform ab, sondern insbesondere von den verfügbaren Erweiterungen im Plattform-Ökosystem. Add-on-Stores bieten Zugang zu spezialisierten Erweiterungen, die die Grundfunktionen der Plattformen ergänzen und an spezifische Unternehmens-bedürfnisse anpassen können.

Um die Eignung einer LCP/NCP für spezielle Anwendungsfälle zu bewerten, entwickelt dieser Beitrag ein Messinstrument zur Erfassung der Komplexität von Add-on-Stores unter Anwendung der Design Science Research Methode (DSR). Die Bewertung basiert auf der Analyse von 1483 Add-ons aus den Add-on-Stores von Microsoft Power Apps, Outsystems und Pega. Dieses Instrument ermöglicht es Unternehmen, die Vielfalt und Tiefe der verfügbaren Add-ons zu erfassen und somit fundierte Entscheidungen darüber zu treffen, ob eine LCP/NCP adoptiert werden

✉ Adrian Abendroth · Benedict Bender
Lehrstuhl für Wirtschaftsinformatik, insb. Prozesse und Systeme, Universität Potsdam, 14482 Potsdam, Deutschland
E-Mail: adrian.abendroth@wi.uni-potsdam.de

Benedict Bender
E-Mail: benedict.bender@wi.uni-potsdam.de

soll. Dieser Beitrag unterstützt somit die Optimierung der Entscheidungsfindung bei der Auswahl von LCP/NCP.

Schlüsselwörter Low-Code · No-Code · LC/NC Plattformen · Marktplätze · Add-ons · Komplexität

A Tool for Analysing the Complexity of Low-Code and No-Code Add-Ons: a Design Science Approach

Abstract In today's fast-moving business world, it is crucial for companies to adapt quickly to changing market conditions in order to remain competitive. Flexibility and customised processes are key success factors that often go beyond the capabilities of standardised application systems. Low-code platforms (LCP) such as Mendix and Outsystems as well as no-code platforms (NCP) such as Bubble offer promising approaches for realising individual adaptations using visual and declarative techniques that replace conventional programming.

Before adopting such a platform, companies should assess how well a specific LCP/NCP can fulfil their individual requirements. The ability of the platforms to address individual needs depends not only on the core functions, but especially on the available extensions in the platform ecosystem. Add-on stores provide access to specialised extensions that complement the basic functions of the platforms and can be adapted to specific business needs.

In order to assess the suitability of an LCP/NCP for specific use cases, this paper develops a measurement tool to capture the complexity of add-on stores using the Design Science Research Method (DSR). The assessment is based on the analysis of 1483 add-ons from the add-on stores of Microsoft Power Apps, Outsystems and Pega. This tool enables organisations to understand the variety and depth of available add-ons and thus make informed decisions about whether to adopt an LCP/NCP. This article thus supports the optimisation of decision-making when selecting LCP/NCP.

Keywords Low-Code · No-Code · LC/NC platforms · Marketplaces · Add-ons · Complexity

1 Einleitung

In der schnelllebigen Geschäftswelt von heute sind Unterscheidungsmerkmale im Wettbewerb von entscheidender Bedeutung für den langfristigen Erfolg. Daher ist es unerlässlich, Alleinstellungsmerkmale und Wettbewerbsvorteile zunehmend in digitalen Prozessen zu verankern. Unternehmen müssen in der Lage sein, sich schnell an veränderte Marktbedingungen anzupassen, um ihre Wettbewerbsfähigkeit zu erhalten. Flexibilität ist ein entscheidender Erfolgsfaktor für das Überleben und die Anpassungsfähigkeit an Kundenbedürfnisse (Teece et al. 2016).

Standardisierte Anwendungssysteme bieten zwar Vorteile bei der Bereitstellung von Best Practices, lassen aber häufig keine Abbildung von individuellen Arbeits-

weisen zu. Dies führt dazu, dass die betriebliche Standardsoftware angepasst werden muss (Balint 2017). Diese Herausforderung wird durch die Tatsache verschärft, dass Prozessexpert:innen oder Fachanwender:innen oft nicht in der Lage sind, die notwendigen Anpassungen selbstständig vorzunehmen. Dies liegt an verschiedenen Restriktionen wie der Bindung an den Systemanbieter (Haddara et al. 2022), fehlende Kompetenzen der Mitarbeiter:innen (Binzer und Winkler 2023), die hohen Kosten für Spezialist:innen (Almutairi et al. 2022) oder fehlende Kompetenzen der Unternehmen für Anpassungen (Davison et al. 2021).

In den letzten Jahren haben sich Low-Code Plattformen (LCP) wie Mendix, Outsystems und No-Code Plattformen (NCP) wie Bubble als technologische Innovationen etabliert, welche diese Herausforderungen adressieren. LCP/NCP stellen eine Zusammenstellung von Tools dar, die sich sowohl für Entwickler:innen als auch Nicht-Entwickler:innen eignen, um Geschäftsanwendungen mit niedrigem Aufwand zu erstellen (Adrian et al. 2020; Bock und Frank 2021; Rymer 2017; Waszkowski 2019) und mittels LCP/NCP Anpassungen an Anwendungssystemen vorzunehmen (Käss et al. 2023). Anstelle herkömmlicher Programmierung nutzen diese Plattformen visuelle und deklarative Techniken wie Drag & Drop (Rymer 2017; Waszkowski 2019), um Benutzeroberflächen, Geschäftslogiken und Datenmodelle zu erstellen (Metrôlho et al. 2019; Sahay et al. 2020; Totterdale 2018). Somit ermöglichen LCP/NCP Entwickler:innen die Erstellung von Anwendungssoftware mithilfe grafischer Benutzeroberflächen und Vorlagen anstelle traditioneller Programmiersprachen (Wang und Wang 2021).

Für Unternehmen stellt sich daher die Frage, ob die Einführung einer LCP/NCP vorteilhaft ist (Käss et al. 2022). In der Phase vor der Adoption (Pre-Adoption) einer Technologie sollte deren Vorteilhaftigkeit evaluiert werden (Karahanna et al. 1999; Wulf et al. 2022). Dazu wird üblicherweise die Zweckmäßigkeit der Adoption unter Berücksichtigung des Mehrwerts der Plattform im Vergleich zu den damit verbundenen Kosten bewertet. In Hinblick auf die Funktionalität sind wiederum die durch LCP/NCP zur Verfügung gestellten Funktionen (vergleichbar mit Standardsoftware) entscheidend (Rokis und Kirikova 2023).

Das mit einer LCP/NCP verbundene Ökosystem bietet Zugang zu spezialisierten Funktionen, welche in Applikationen, sogenannten Add-ons, gebündelt sind und die Grundfunktionalität, die eine LCP/NCP bereitstellt, erweitern. Add-on-Stores wie z. B. Microsoft AppSource bieten mehrere tausend Add-ons welche Unternehmen beziehen können (Lourenço et al. 2023). Unternehmen können nach Add-ons, die ihre Unternehmensanforderungen erfüllen, suchen und diese verwenden. Die Add-ons fördern die Modularität und Wiederverwendbarkeit von Code, was wiederum die Entwicklungseffizienz steigert und die Wartbarkeit erleichtert (Bender 2021). Somit können Effizienzsteigerungen für Unternehmen geschaffen werden (Käss et al. 2023). Die auf einer LCP/NCP verfügbaren Add-ons sind also ein wesentliches Element zur Beurteilung der Eignung einer Plattform für den Unternehmenseinsatz. Die Anzahl und die Qualität der in den Add-on Stores verfügbaren Erweiterungen geben Aufschluss über die Flexibilität der Plattform, spezielle Anforderungen zu erfüllen und die Grundfunktionalität zu erweitern.

Im Markt sind zahlreiche LCP/NCP verfügbar aus denen Unternehmen eine Auswahl treffen müssen. In diesem Kontext stellt sich für Unternehmen nicht nur die

Frage der Vorteilhaftigkeit der Technologie, sondern insbesondere auch der funktionale Mehrwert, den eine spezifische LCP/NCP bietet.

Die **Forschungsfrage** ist demnach:

RQ: Welche LCP/NCP eignen sich am besten für Unternehmen, angesichts ihrer spezifischen Funktionsanforderungen?

Um die Forschungsfrage zu beantworten, ist eine Bewertung verschiedener LCP/NCP erforderlich. Für Unternehmen gestaltet sich diese Bewertung jedoch oft als schwierig, da unklar bleibt, welche spezialisierten Anwendungsfälle von den verschiedenen Plattformen abgedeckt werden können. Abstrahiert man vom Vergleich der Grundfunktionen einer LCP/NCP, bildet die Analyse der zugehörigen Stores und der dort verfügbaren Add-ons einen möglichen Ansatz. Gibt es viele Add-ons in verschiedenen Bereichen, d. h. ist die „Komplexität" des Add-on-Stores hoch, ist die Chance tendenziell höher, auch viele verschiedene Unternehmensanforderungen mit der LCP/NCP abbilden zu können.

Somit könnte ein detailliertes Komplexitätsprofil eines Add-on-Stores einer LCP/NCP Aufschluss über ihre Fähigkeit geben, individuelle Unternehmensanwendungsfälle zu realisieren.

Hierfür entwickelt der vorliegende Beitrag ein Messinstrument, mit dem die Komplexität von LCP/NCP-Add-on-Stores quantifiziert werden können. Hierzu werden die Beschreibungen der Add-ons ausgewertet. In einem zweiten Schritt wird dieses Instrument auf drei ausgewählte Add-on-Stores viel genutzter LCP und die dortigen Add-on-Beschreibungen angewendet, um einen umfassenden und aktuellen Marktüberblick zu erhalten.

In Kap. 2 wird die relevante Forschung dargestellt. Anschließend wird in Kap. 3 das methodische Vorgehen zur Entwicklung des Messinstruments erläutert, das dann in Kap. 4 entwickelt wird. Kap. 5 demonstriert die Anwendung des Messinstruments am Beispiel von drei Add-on Stores inklusive Auswertung. Den Abschluss bilden die Evaluierung (Kap. 6) und die Diskussion der Ergebnisse (Kap. 7).

2 Relevante Forschung

Unternehmen stehen vor der Herausforderung, zu beurteilen, ob ihre bestehende Systemlandschaft ausreichend flexibel ist, um neue Anpassungen an ihren Anwendungssystemen vorzunehmen. In diesem Kontext wird zunehmend das Thema LCP/NCP von Marktforschungsunternehmen wie Gartner und Forrester diskutiert. Diese Plattformen sind als neuer Trend auf dem Markt präsent und versprechen innovative Ansätze zur Anpassung von Softwarelösungen (Bratincevic 2024; Gartner 2022).

Individuelle, wettbewerbsdifferenzierende Prozesse eines Unternehmens lassen sich nicht in Standardsoftware abbilden (Lokuge und Sedera 2017; Rajagopal 2002) oder verursachen hohe Kosten, wenn Anpassungen von externen Partnern durchgeführt werden (Fryling 2010; Hitt et al. 2002; Parthasarathy und Sharma 2017). Mangelnde Flexibilität und Anpassungsfähigkeit kann zu einem Wettbewerbsnachteil führen (Lokuge und Sedera 2017). Deshalb sind Unternehmen daran interessiert,

Anpassungen an Anwendungssystemen kurzfrisitig, effizient und kostengünstig umzusetzen.

LCP/NCP bieten als vielversprechende Alternative einen neuartigen Ansatz für das Spannungsfeld der Anpassungsprogrammierung. Diese Plattformen ermöglichen es Unternehmen, standardisierte Software durch individuelle Prozesse zu ergänzen, die wettbewerbsdifferenzierende Merkmale darstellen (Elshan et al. 2023; Käss et al. 2023). Durch die Nutzung von LCP/NCP können Unternehmen ihre Flexibilität verbessern und ihre Anpassungsfähigkeit an die Anforderungen des Marktes steigern, ohne dabei auf die Vorteile standardisierter Software zu verzichten (Brünninghaus und Röcker 2022).

LCP/NCP sind Plattformen, die vordefinierte Funktionen und Komponenten anbieten, die ohne weitere Konfiguration verwendet werden können. Geschäftsanwendungen lassen sich durch visuelle Techniken und Drag & Drop einfach zusammenstellen. In diesen visuellen Umgebungen können sowohl vordefinierte als auch eigene Komponenten erstellt, verwendet und miteinander kombiniert werden (Bock und Frank 2021; Hirzel 2023; Woo 2020). Darüber hinaus sind auch Code-Anpassungen innerhalb der einzelnen Komponenten möglich, was zusätzliche Flexibilität und Individualisierungsmöglichkeiten bietet (Cabot 2020; Lethbridge 2021).

NCP stellt eine spezifische Untergruppe von LCP dar, die sich auf einer höheren Abstraktionsebene befindet (Curty et al. 2023). Auf diesen Plattformen ist es nicht möglich, Anpassungen über Codeänderungen zu schreiben, sondern nur die verfügbaren Komponenten zu nutzen (Cabot 2020). Diese Einschränkung führt zu weniger Gestaltungsfreiheit für die zu entwickelnden Applikationen (Cabot 2020). Während LCP für komplexere Anwendungen genutzt werden, werden NCP bei simpleren Anwendungen verwendet. NCP sind darüber hinaus intuitiver und die Nutzung der bereitgestellten Tools ist einfacher (Picek 2023).

Durch die Nutzung von LCP/NCP können Unternehmen ihre Flexibilität verbessern und ihre Anpassungsfähigkeit an die Anforderungen des Marktes steigern, ohne dabei auf die Vorteile standardisierter Software zu verzichten (Brünninghaus und Röcker 2022).

Ein wesentliches Differenzierungsmerkmal der LCP/NCP-Plattformen ist ihr verbundenes Ökosystem in Form von Add-on-Stores.

Plattformanbieter offerieren im Wesentlichen eine Grundfunktionalität (auch Kern genannt), welche für den Massenmarkt konzipiert ist und sich in ihren Eigenschaften nicht wesentlich von den Funktionen der konkurrierenden Anbieter unterscheiden.

In der Konsequenz können sich Plattformanbieter nur durch das Angebot spezialisierter Funktionen und Anwendungen von ihren Mitbewerbern differenzieren (Cennamo und Santalo 2013).

Der Store bietet Drittanbietern die Möglichkeit, spezialisierte Funktionen und Anwendungen bereitzustellen, welche die Grundfunktionalität der Plattform ergänzen. Die entsprechenden Anwendungen können von den Fachanwender:innen bezogen werden (Haile und Altmann 2016; Olleros 2008). Indem Plattformen bessere und vielfältigere Funktionen als konkurrierende Plattformen zusammen mit den Drittanbietern anbieten, können sich die Plattformbetreiber differenzieren und dadurch mehr Nutzer anziehen (Nikou et al. 2014).

Die Ausrichtung der Add-on-Stores variiert stark zwischen den verschiedenen Plattformbetreiber (Ghazawneh und Henfridsson 2015), was dazu führt, dass diese Add-on-Stores jeweils sehr individuell ausgeprägt sind. Die Gestaltung und Verwaltung dieser Add-on-Stores, einschließlich der Offenheit der Plattform, des Zugangs für Drittanbieter, der Qualitätskontrolle und der Tools für Drittentwickler:innen, liegt vollständig in den Händen des jeweiligen Plattformanbieters (Bender 2021). Dies führt dazu, dass die Add-on-Stores der unterschiedlichen LCP/NCP-Plattformen erhebliche Unterschiede aufweisen, insbesondere hinsichtlich der Anzahl der verfügbaren Add-ons und der von ihnen angebotenen Funktionen (Boudreau 2010).

Die Qualität und Vielfalt der verfügbaren Add-ons sind entscheidende Faktoren, die die Fähigkeit der Plattform zur Anpassung an besondere Anforderungen und zur Erweiterung ihrer Grundfunktionen beeinflussen können. Deshalb sollten Unternehmen vor der Entscheidung für die Adoption einer spezifischen LCP/NCP-Plattform die verfügbaren Add-on-Stores der Plattformen intensiv analysieren, wodurch Unsicherheiten bei der Auswahl der passenden Plattform verringert werden können (Müller et al. 2011; Song et al. 2013).

Ein Ansatz zur Analyse der Add-on-Stores könnte darin bestehen, die Komplexität der dort verfügbaren Add-ons zu bestimmen. Traditionelle Ansätze zur Komplexitätsmessung in der Softwareentwicklung basieren auf der Analyse von Quellcode (Lake und Cook 1994), der in ausreichendem Umfang verfügbar sein muss. Diese Analysen umfassen unter anderem die Untersuchung von Kontrollflüssen (McCabe 1976) Informationsflüssen (Lake und Cook 1994) sowie die Analyse von Operanden und Operatoren (Halstead 1977). Während diese Metriken für textbasierte, programmierzentrierte Ansätze entwickelt wurden, fehlen Methoden, die die Komplexität aus nicht-textuellen, insbesondere visuellen Beschreibungen ableiten.

Da LCP/NCP-Plattformen visuelle und deklarative Techniken zur Softwareentwicklung nutzen, sind traditionelle Messmethoden, die auf Codeanalysen basieren, nicht geeignet, um die spezifischen Anforderungen und Möglichkeiten dieser neuen Entwicklungsansätze abzubilden. Dies verdeutlicht den Bedarf an neuen Metriken, die die tatsächliche Komplexität und den Aufwand in LCP/NCP genauer erfassen können.

Aufgrund der Besonderheiten von LCP/NCP ist es notwendig, spezifische Messinstrumente zu entwickeln, die die Komplexität dieser Plattformen im Kontext ihres Ökosystems erfassen können. Ein zentrales Differenzierungsmerkmal dieser Plattformen ist ihr verbundenes Ökosystem in Form von Add-on-Stores.

Ein Messinstrument, das die Vielfalt und Integrationstiefe der verfügbaren Add-ons berücksichtigt, würde es Unternehmen ermöglichen, die Plattformen umfassender zu bewerten. Dies würde Unternehmen helfen, fundierte Entscheidungen darüber zu treffen, welche Plattform ihren spezifischen Anforderungen am besten entspricht und welches Add-on-Angebot ihnen den größten Mehrwert bietet. So könnten Unsicherheiten bei der Wahl einer Plattform verringert und die Passgenauigkeit der gewählten Lösung für individuelle Geschäftsanforderungen erhöht werden.

Tab. 1 Hauptphasen mit Prozessschritten der DSR-Methode mit zusätzlichen Methoden

Hauptphase	Prozessschritt	Methode	Kapitel
Problemidentifikation	Problemidentifikation und Motivation	Sichtung relevanter Forschung	Kap. 1
	Ziel der Lösung	Festlegung der Lösungsziele	Kap. 2
Lösungsentwurf	Design und Entwicklung	Artefaktentwicklung (Komplexitätsmessung) nach MacKenzie et al. (2011)	Kap. 4
Evaluierung	Demonstration	Vorgehen zur Datensammlung angeleht an Feldman und Sanger (2007, S. 15)	Kap. 5
	Evaluierung	Klassifikationsevaluierung	Kap. 6
	Kommunikation	Veröffentlichung der Ergebnisse	Kap. 7–8

3 Methodisches Vorgehen

In der Wirtschaftsinformatik stellt das Design Science Research (DSR) eine etablierte Forschungsmethode dar, welche der strukturierten Entwicklung von Artefakten dient. Diese basieren auf wissenschaftlichen Theorien und zielen auf die Lösung realer Probleme ab (Becker et al. 2020). Zur wissenschaftlich anerkannten Entwicklung des Messinstruments orientieren sich die Autoren an Vorgehensmodellen für DSR. In der Literatur werden zahlreiche DSR-Prozesse vorgeschlagen, wobei der Ansatz von Peffers et al. (2007) der am weitesten verbreitete und am häufigsten angewendete ist. Diesen Ansatz nutzen die Autoren, um ein konkretes Messinstrument zu entwickeln. Grundsätzlich lassen sich DSR-Prozesse in drei Hauptphasen unterteilen: Problemidentifikation, Lösungsentwurf und Evaluation (Offermann et al. 2009).

Die Phase der Problemidentifikation wird durch die Einleitung und den theoretischen Hintergrund abgedeckt. Es wird ein Komplexitätsmessinstrument entwickelt, das auf Add-on-Beschreibungen beruht. Der Vorteil dieser Vorgehensweise besteht darin, dass Add-on-Beschreibungen spezifische Informationen über die Erweiterungen liefern, die für die Bewertung der Komplexität relevant sind. Das Messinstrument adressiert die Problematik, dass bisher keine geeignete Metrik vorhanden ist. Bisherige Skalen beruhen auf dem verfügbaren Quellcode. Der Lösungsentwurf wird in Kap. 4 beschrieben, während die Demonstration in Kap. 5 stattfindet. Die Evaluierung der Messskala erfolgt in Kap. 6. Die Kommunikation wird in den Kap. 7 und 8 dargestellt (Tab. 1).

4 Design und Entwicklung einer Komplexitätsausprägungsskala für LCP/NCP

Um ein standardisiertes Messinstrument für die Bewertung der Komplexität von LCP/NCP Add-Ons zu entwickeln, orientiert sich die Vorgehensweise an dem Ansatz zur Entwicklung eines Komplexitätskonstrukts von MacKenzie et al. (2011). Zunächst wird das Konzept der Komplexität aus der vorhandenen Literatur im Bereich der Softwaremetriken abgeleitet. Anschließend erfolgt die Festlegung einer Skala für die Ausprägung der Komplexität, wobei Funktionen und ihre Schwierig-

keiten bei der Implementierung aus der bestehenden LCP/NCP Literatur gesammelt werden. Darauf aufbauend werden die LCP/NCP-Funktionen als Anforderungen hergeleitet und entsprechend auf einer Skala von einfach bis schwierig über Expert:innen bewertet.

4.1 Bestimmung des Komplexitätskonstrukts

Charakteristisch für die Komplexität von Software ist die Verständlichkeit eines Programms oder Systems für Entwickler:innen. Dies umfasst Aspekte wie das Software-Design, die Anzahl der Komponenten und ihre Beziehungen zueinander (Cho et al. 2001), den mentalen Aufwand für das Verständnis der Software (Basili 1988; Zuse 2019) sowie die Wartung und Testung (Basili 1988; Schneider et al. 1981; Zuse 2019). Aspekte wie Modularität und Wiederverwendbarkeit kann das Verständnis fördern (Xiang et al. 2019). Für die Bewertung der Softwarekomplexität werden spezialisierte Metriken genutzt. Es ist jedoch schwierig, die Komplexität allein anhand einer Skala zu erfassen (Zuse 2019). Diese Metriken dienen dazu, die Software anhand spezifischer Qualitätsmerkmale zu vergleichen, wobei Eigenschaften des Programmiercodes gezählt werden (Sonnleithner und Zoitl 2020; Weyuker 1988). Dabei liegt der Fokus nicht direkt auf der Quantifizierung der Komplexität selbst, sondern auf der Bewertung bestimmter Programmeigenschaften, die als Ursache für Komplexität angesehen werden können (Lake und Cook 1994).

Gängige Metriken zur Bewertung der Softwarekomplexität umfassen die Anzahl der Codezeilen (Rosenberg, 1997), die Größe basierend auf Halsteads Zählungen von Operanden und Operatoren (Halstead 1977), die Anzahl der Kontrollflüsse und ihrer Verzweigungen nach McCabe (McCabe 1976) sowie Metriken zum Informationsfluss, die den Datenaustausch und Datenabrufe messen (Lake und Cook 1994). Diese Metriken sind zum Großteil nur auf text-basierten Programmierungen anwendbar und nicht auf grafisches Programmieren bezogen (Fischer et al. 2021). Ruan et al. (2017) schlagen für das blockbasierte Programmieren vor, die Abzweigungen in den Blöcken nachzuvollziehen, um die Komplexität zu messen. Diese Empfehlung wird auch von Ono et al. (2024) unterstützt, die feststellen, dass Abzweigungen (die auf Programme mit Wiederholungen und bedingten Verzweigungen hinweisen) ein wirksames Mittel sind, um die Komplexität beim blockbasierten Programmieren zu messen. Aus den genannten Metriken wurden Komplexitätsaspekte abgeleitet und auf LCP/NCP übertragen (Tab. 2).

Weitere Indikatoren aus dem LCP/NCP-Bereich, die auf Komplexität für Entwickler:innen hinweisen sind Integrationen (Alamin et al. 2023; Käss et al. 2022) sowie im speziellen die Integration von APIs (Alamin et al. 2021) sowie Themen im Zusammenhang mit der Plattformadoption, wie beispielsweise die Verwendung von Nachrichtenwarteschlangen, plattformbezogene Abfragen oder interaktive Berichten (Alamin et al. 2023).

4.2 Darstellung des Komplexitätskonstrukts

Die Softwarekomplexität bei LCP/NCP lässt sich in Analogie zur Softwarekomplexität wie folgt definieren: Der Grad der Schwierigkeit, eine durch eine LCP/NCP-

Tab. 2 Abgeleitete Komplexitätsaspekte übertragen auf LCP/NCP

Komplexitätsaspekt	Beschreibung	Quelle	LCP/NCP-Kontext
Codezeilen	Dieser Aspekt umfasst die Menge an Code, typischerweise gemessen in der Anzahl der Codezeilen. Diese Kennzahl gibt Aufschluss über die Größe und den Umfang eines Softwareprojekts. Mehr Codezeilen können auf eine höhere Komplexität hinweisen, da mehr Logik, Funktionalität und möglicherweise auch mehr Abhängigkeiten und Schnittstellen im Code enthalten sind	Lake und Cook (1994)	Generierter Code durch LCP/NCP basierend auf genutzten Komponenten
Operanden	Operanden beziehen sich auf die Variablen und Konstanten, die innerhalb des Codes verwendet werden. Er umfasst die Definition, Deklaration und Nutzung von Variablen und Konstanten, die in verschiedenen Teilen des Programms vorkommen	Halstead (1977)	Variablen, Konstanten, Eingaben aus Benutzeroberflächen, Datenfelder usw., die in den visuellen Blöcken verwendet werden
Operatoren	Dieser Aspekt konzentriert sich auf die Vergleichsoperatoren, die im Code verwendet werden. Vergleichsoperatoren sind essenziell für die Entscheidungsfindung innerhalb eines Programms, da sie Bedingungen und Vergleiche ermöglichen, die den Programmfluss steuern	Halstead (1977)	Visuelle Abzweigungen in Blöcken
Kontrollfluss	Der Kontrollfluss bezieht sich auf Reihenfolge und Struktur, in denen Anweisungen abgearbeitet werden (Ausführungspfade). Eine weitverbreitete Metrik ist die zyklomatische Komplexität nach McCabe	McCabe (1976)	Visualisierte Workflows, Verzweigungspunkte und Verbindungen
Informationsfluss	Der Informationsfluss erfasst die Anzahl der Datenzugriffe innerhalb eines Programms. Die Komplexität großer Programme wird durch die Menge der geteilten Daten und die Anzahl der Aufrufe zwischen verschiedenen Modulen oder Komponenten gemessen. Ein hoher Informationsfluss kann auf eine intensivere Kommunikation und Datenabhängigkeit zwischen den Programmteilen hinweisen	Lake und Cook (1994)	Eigene oder LCP/NCP spezifische Komponenten die aufeinander zugreifen

Plattform erstellte Softwareanwendung zu verstehen, zu modifizieren und zu erweitern, stellt ein geeignetes Kriterium zur Messung der Komplexität dar. Die Messung dieser Komplexität kann anhand verschiedener Dimensionen erfolgen, darunter die Anzahl der generierten Codezeilen, die Anzahl und Art der Operanden und Operatoren, die Struktur des Kontrollflusses, die Menge und Komplexität der Informationsflüsse sowie spezifische Integrationen und plattformspezifische Funktionen.

Das Konstrukt umfasst notwendige und hinreichende Merkmale, die sowohl gemeinsame als auch einzigartige Aspekte berücksichtigen. Zu den gemeinsamen Merkmalen zählen Elemente, die in nahezu jeder LCP/NCP-Awnendung vorhanden sind, wie Codezeilen und Kontrollfluss. Einzigartige Merkmale beinhalten spezifische Integrationen und plattformspezifische Funktionen, die nicht in jeder Anwendung vorkommen. Die Breite und Inklusivität des Konstrukts spiegelt sich in der Vielzahl der Aspekte wieder, die sowohl allgemeine als auch spezifische Merkmale der Softwarekomplexität abdecken. Das Konstrukt der Softwarekomplexität bleibt stabil über die Zeit, auch wenn sich die Technologie weiterentwickelt. Es gilt für verschiedene Arten von LCP/NCP-Add-ons und ist über Fälle hinweg anwendbar.

4.3 Erstellung von Ausprägungen der Komplexität

Die gegenwärtig verfügbaren Softwaremetriken erweisen sich als unzureichend, um die Spezifik wie z. B. grafische Spezifikation mittels Drag-&-Drop-Editoren von LCP/NCP adäquat zu erfassen, insbesondere wenn der zugrunde liegende Code nicht verfügbar ist. Die Messung der Komplexität von LCP/NCP-Add-ons erfordert die Entwicklung einer geeigneten Skala.

In der wissenschaftlichen Literatur findet sich eine Einteilung der Funktionen im LCP/NCP-Bereich in die Kategorien „einfach“ und „schwer“ hinsichtlich des Schwierigkeitsgrads (Alamin et al. 2021, 2023; Binzer und Winkler 2023; Bernsteiner et al. 2022). Eine weitere Abstufung ist nicht ersichtlich. Diese dichotome Kategorisierung erweist sich als unzureichend, um die vielfältigen Ausprägungen von Komplexität adäquat abzubilden.

Zur Lösung des vorliegenden Problems wurde die dichotome Kategorisierung aus der Literatur in eine Komplexitätsskala mit fünf Ausprägungen erweitert. Im ersten Schritt erfolgte eine Zusammenstellung der aus der Literatur bekannten Funktionen, welche hinsichtlich ihres Schwierigkeitsgrades als „einfach“ oder „schwer“ kategorisiert wurden. Basierend auf diesen Einordnungen wurden entsprechende Anforderungen formuliert (siehe Tab. 6). Die Bewertung erfolgt auf Basis von Analysen und vorherigen Erfarhungen mit LCP/NCP-Technologien. Dies bildet die Grundlage für die Entwicklung des Kodierungsleitfasdens.

Mit Hilfe des vorliegenden Kodierleitfadens können die Beschreibungen der Add-ons in Ihrer Gesamtkomplexität kategorisiert werden. Jedem Add-on wird eine Gesamtkomplexität zwischen 1 (sehr gering) und 5 (sehr hoch) komplex gegeben.

Es besteht die Möglichkeit, dass ein Add-on mehrere der abgeleiteten Anforderungen aus dem Kodierleitfaden beinhaltet. Bei einem Add-on, welches überwiegend komplexere Funktionalität bereitstellt, wird eine höhere Gesamtkomplexität zugewiesen. In diesem Zusammenhang ist von entscheidender Bedeutung, dass die identifizierten Anforderungen (Komplexitätsaspekte) einen wesentlichen Bestand-

Tab. 3 Beispiele für Gesamtkomplexität

Gesamtkomplexität	Einordung	Beispiele von zugehörenden Komplexitätsausprägungen
Sehr geringe Komplexität (1)	Erstellung einfacher Masken mit Grundfunktionen der LCP/NCP mit wenig logischen Operationen. Konfiguration in LCP/NCP einfach abbildbar	Sammlung von einfachen Templates, Reports und Checklisten Auswertung in Form von Dashboards
Geringe Komplexität (2)	Add-ons mit Realisierungsanforderungen und Aufwand in einem abgegrenzten Bereich mit einfachen Funktionen, die mittels einfachen logischen Operationen abgebildet werden können	Add-ons in Form von CRUD (Create, Read, Update, Delete) Zeiterfassung, einfache CRM- und Servicemanagement Funktionen
Mittlere Komplexität (3)	Diese Add-ons weisen höhere Anzahl an Verzweigungen und Bedienungen auf in einem abgegrenzten Bereich auf. Es sind höhere, logische Kenntnisse erforderlich	Add-ons, welche komplexere Bedingungen, Logiken oder Abzweigungen beinhalten Komplexere, verschachtelte Datenbankabfragen
Hohe Komplexität (4)	Bei den Add-ons sind grundlegende Programmier-, Datenbank- und Modellierungskenntnisse notwendig, um Logiken zu verstehen und kleinere Anbindungen an die Add-ons zu ermöglichen	API-Calls zu weiteren Systemen Integration von Drittsystemen wie Maps für die keine Standardintegration vorliegt
Sehr hohe Komplexität (5)	Bei den Add-ons sind umfangreiche Programmierkenntnisse in spezifischen Programmiersprachen notwendig. Es ist nicht möglich diese Add-ons mittels der vordefinierten Komponenten der LC zu erstellen. Das beinhaltet u. a. das Erstellen komplexer Algorithmen. Auch komplexe Add-ons, welche Zusatzwissen benötigen, fallen in die Kategorie	Abbildung individueller Algorithmen Komplexe Integrationen zu anderen Systemen Spezifische Kenntnisse wie .NET, Java, SOAP erforderlich Externe Systeme mit Vielzahl Funktionen und Schnittstellen, in denen ein hohes logisches Verständnis notwendig ist

teil des Add-ons darstellen. Eine Betrachtung der identifizierten Anforderungen darf nicht isoliert, sondern muss stets in Relation zum gesamten Add-on erfolgen. Ein Add-on, welches eine komplexe Anforderung umfasst, diese jedoch lediglich einen geringen Teilbereich betrifft, kann dennoch eine niedrigere Gesamtkomplexität aufweisen. Die Einschätzung obliegt der kodierenden Person. Tab. 3 zeigt exemplarisch Beispiele für die Gesamtkomplexität von Add-ons.

5 Demonstration der Gesamtkomplexitätsskala

Zur Anwendung der Gesamtkomplexitätsskala werden Daten in Form von Beschreibungen benötigt. Dazu wird sich an den vierstufigen Prozess gemäß Feldman und

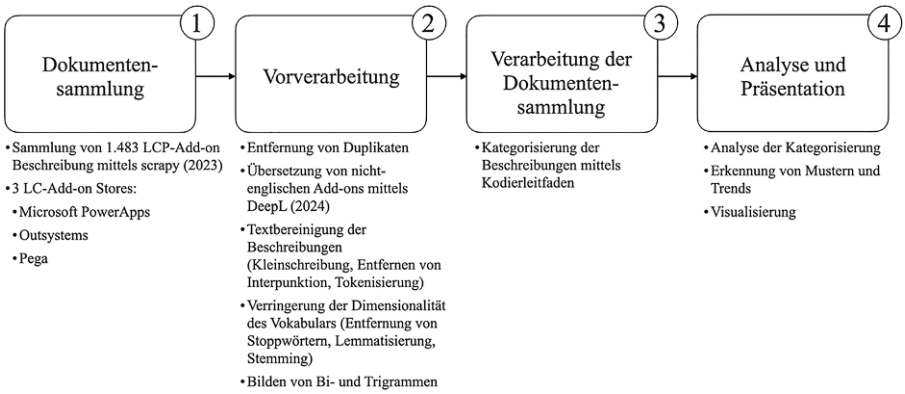

Abb. 1 Schritte der LCP-Add-on Store Analyse angepasst an Feldman und Sanger (2007, S. 15)

Sanger (2007, S. 15) orientiert (Abb. 1). Zuerst wurden die Daten aus den LCP-Add-on Stores mittels Web Scraping gesammelt. Anschließend erfolgte in einem zweiten Schritt die Vorverarbeitung, bei der die gesammelten Daten bereinigt wurden. Im dritten Schritt, der Bearbeitung der Dokumentensammlung, erfolgte die manuelle Kategorisierung der Daten auf Basis des erstellen Kodierleitfadens. Im letzten Schritt wurden die Daten auf Muster und Trends der Kategorien analysiert und visualisiert.

5.1 Dokumentensammlung

Für die Analyse wurden drei relevante Anbieter von LCP mit Add-on-Stores identifiziert, welche öffentlich zugänglich sind. Microsoft Power Apps, Pega und Outsystems wurden ausgewählt, um die Beschreibungen ihrer LCP-Add-on Stores qualitativ zu analysieren. Die Beschreibungen der Add-on Stores wurden mithilfe des Open-Source-Web-Crawling- und Web Scraping-Python-Frameworks scrapy in Version 2.11.0 gesammelt (scrapy 2023). Für jedes LCP-Add-on wurden der Titel sowie die textuelle Beschreibung gesammelt. Die LCP-Add-on Stores wurden ausgewählt, da die Beschreibungen ohne Zugang frei verfügbar sind und die Struktur der einzelnen Webseiten konsistent ist, um die Daten zu erfassen. Im Zeitraum von Mitte Januar bis Ende Januar 2024 wurden insgesamt 1483 Add-on Beschreibungen aus den genannten LCP-Add-on Stores gesammelt (Tab. 4).

Tab. 4 Verteilung von Datenpunkten durch LCP-Add-on Stores

LCP-Store	Anzahl	Referenz
Microsoft Power Apps AppSource	1084 (73 %)	Microsoft (2024)
Outsystems	231 (16 %)	Outsystems (2024)
Pega	168 (11 %)	Pega (2024)

Abb. 2 Verteilung der Gesamtkomplexität

5.2 Vorverarbeitung

Zuerst wurden die einzelnen Beschreibungen auf Duplikate überprüft ($n = 0$). Danach wurden Beschreibungen auf Vollständigkeit überprüft. Im Nachgang wurden Beschreibungen mit DeepL (2024) übersetzt, die nicht in der deutschen oder englischen Sprache verfügbar waren ($n = 52$).

Auf Basis der gesammelten 1483 LCP-Add-on Store Daten wurde die entwickelte Gesamtkomplexitätsskala angewandt, um die Forschungsfrage zu beantworten. In Tab. 7 werden konkrete Beispiele der Daten auf eine Gesamtkomplexitätskategorie sowie die verbundene Kodierung gezeigt.

5.3 Ergebnisse der Demonstration

Die Gesamtkomplexität, welche das Resultat der Klassifizierung darstellt, weist lediglich einen geringen Anteil an mittelkomplexen Add-ons auf, welcher 115 (7,6 %) beträgt. Ein knappes Drittel der Add-ons lässt sich als gering komplex (454; 30,6 %) oder stark komplex (320; 21,6 %) kategorisieren. In Bezug auf die Gesamtkomplexität lässt sich festhalten, dass 511 (34,5 %) als gering und 83 (5,6 %) als sehr gering komplex bewertet werden können. Somit lassen sich 594 (40,1 %) dem gering komplexen Grad und 774 (52,2 %) dem hoch komplexeren Grad zuordnen (Abb. 2).

Eine ähnliche Verteilung der Gesamtkomplexität lässt sich bei Betrachtung der LCP-Add-on Stores feststellen. Bei höheren Komplexitäten lassen sich jedoch signifikante Abweichungen in der Verteilung der Gesamtkomplexität beobachten. Bei Outsystems fallen beispielsweise 38,5 % in die Kategorie der sehr hohen Gesamtkomplexität, während es bei Pega 25,6 % und bei Microsoft AppSource 17,3 % sind. Wenn man die Verteilung der hohen Gesamtkomplexität (4 und 5) sowie der geringeren Geasmtkomplexität (1 und 2) betrachtet, ist die Verteilung insgesamt ähnlich (Tab. 5).

Für den LCP-Store des Microsoft Power Apps AppSource konnten zusätzlich die Kategorien der jeweiligen Add-ons erfasst werden. Diese Kategorien werden von Microsoft festgelegt, wobei jedes Add-on nur einer Kategorie zugeordnet werden

Tab. 5 Gesamtkomplexitätsverteilung sortiert nach LCP-Add-on Stores

LCP-Store	Sehr geringe Gesamtkomplexität (5,6 %)	Geringe Gesamtkomplexität (34,5 %)	Mittlere Gesamtkomplexität (7,6 %)	Hohe Gesamtlomplexität (30,6 %)	Sehr hohe Gesamtkomplexität (21,6 %)	Gesamt
Microsoft Power Apps AppSource	52 (4,8 %)	389 (35,9 %)	97 (8,9 %)	358 (33 %)	188 (17,3 %)	*1084*
OutSystems	25 (10,8 %)	61 (26,4 %)	4 (1,7 %)	52 (22,5 %)	89 (38,5 %)	*231*
Pega	6 (3,6 %)	61 (36,3 %)	14 (8,3 %)	44 (26,2 %)	43 (25,6 %)	*168*
Gesamt	*83*	*511*	*115*	*454*	*320*	*1483*

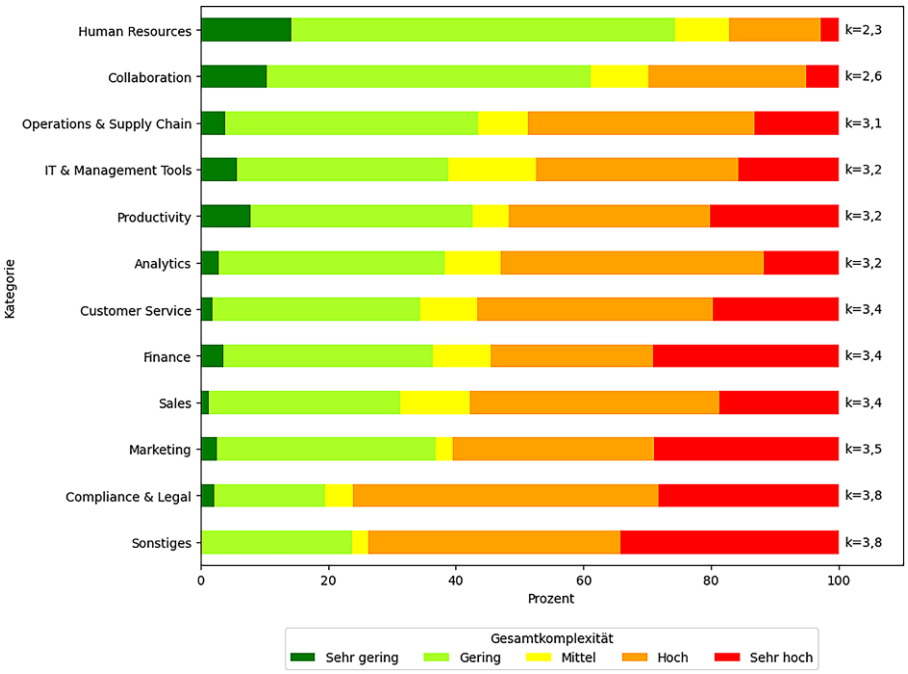

Abb. 3 Verteilung der Gesamtkomplexität und Komplexitätsgrads nach Kategorien aus dem MS Power Apps AppSource

kann. Die am häufigsten verwendete Kategorie im Datensatz ist Sales ($n = 230$), gefolgt von Customer Service ($n = 157$) und IT & Management Tools ($n = 139$).

Wenn man die einzelnen Kategorien in ihrer prozentualen Verteilung betrachtet, zeigen sich merkliche Unterschiede zwischen den Kategorien. Diejenigen mit einem hohen Anteil an Add-ons mit niedriger Gesamtkomplexität sind Human Resources mit 60 %, gefolgt von Collaboration mit 50,6 % (siehe auch Abb. 3 und Tab. 8).

Die prozentuale Verteilung der Gesamtkomplexität in den Kategorien wurde unter Berücksichtigung des durchschnittlichen Komplexitätsgrads (k) auf einer Skala von eins bis fünf berechnet, wobei die prozentuale Verteilung als Gewichtung diente. Auffällig ist, dass die Kategorie „Sonstige“ (k = 3,8) am komplexesten ist. Diese Kategorie umfasst die vier Unterkategorien Geolocation, AI + Machine Learning, Internet of Things und Compliance & Legal, von denen jede einzelne mit weniger als 20 Add-ons vertreten war und daher unter „Sonstige“ zusammengefasst wurde. Die zweitkomplexeste Kategorie ist Compliance & Legal (k = 3,8). Im Gegensatz dazu weisen die Kategorien Human Resources (k = 2,3), Collaboration (k = 2,6) und Operations & Supply Chain (k = 3,1) niedrigere Gesamtkomplexitäten auf.

Ausgehend vom Komplexitätsgrad k wurden die beiden Kategorien mit geringem Komplexitätsgrad, Human Resources und Collaboration, sowie die beiden Kategorien mit hohem Komplexitätsgrad, Marketing und Compliance & Legal, näher betrachtet, um ein besseres Verständnis für die beobachteten Unterschiede zu gewinnen. Die Kategorie „Sonstiges“ wurde nicht einbezogen, da sie zu viele un-

terschiedliche Applikationen umfasst. Aus den vier genannten Kategorien wurden Subkategorien gebildet, die die jeweiligen Add-ons genauer beschreiben.

Bei der Betrachtung der Add-on-Beschreibungen für Human Ressource fällt auf, dass diese Funktionen wie die Buchung von Schreibtischen und Räumen, die Protokollierung von (Covid)-Testprotokollen, die Nominierung von Mitarbeitenden oder kleinere Zeiterfassungstools beinhalten. Die Kategorie Collaboration umfasst zusätzlich Ideenfindungs- und Anerkennungsaddons.

Betrachtet man die einzelnen Add-ons aus der Kategorie Marketing, so wird deutlich, dass der Fokus auf komplexen Marketing-Suites liegt, die gebündelt vielseitige Funktionen anbieten. Zudem gibt es eine größere Anzahl an Customer Relationship Management Add-ons und Integrationen zu E-Mail-Marketing-Programmen.

Für die Kategorie Compliance & Legal sind umfassende Applikationen verfügbar, die Qualitätsmanagement-Add-Ons, Gesundheitsaddons oder Add-Ons, die ISO-Normen beinhalten.

6 Evaluierung

Die Evaluierung des Kodierleitfadens erfolgt anhand der Methode der intersubjektiven Übereinstimmung. Diese Methode dient der Messung der Übereinstimmung zwischen den Kodierergebnissen verschiedener Bewerter. Damit wird festgestellt, inwiefern der Leitfaden unabhängig von individuellen Interpretationen konsistent und anwendbar ist und bildet somit ein Maß für die Qualität und Anwendbarkeit des Kodierungsprozesses darstellt.

Für die Evaluation wurden die 1483 gesammelten Add-on-Beschreibungen anhand des erstellten Kodierleitfadens manuell kodiert. Jede Beschreibung wurde einzeln gelesen und einer Komplexitätskategorie von 1 (sehr gering) bis 5 (sehr hoch) komplex zugeordnet. Der Kodierleitfaden diente als Hilfestellung, um die Komplexität der Entwicklung eines Add-ons anhand der Beschreibung zu bestimmen.

Ein Hauptcodierer, der die Basis für das Messinstrument entwickelt hatte, klassifizierte alle Add-ons. Die vorgenommenen Klassifizierungen dienten als Basis für die nachfolgende Analyse des Datensatzes. Zur Evaluierung der Reliabilität des Messinstruments wurde eine Zufallsauswahl von 100 Add-on-Beschreibungen aus den LCP-Add-on Stores getroffen. Die Stichprobe wurde ebenfalls anhand des Kodierleitfadens von zwei unabhängigen Forschern mit Expertise im Bereich LCP/NCP klassifiziert.

Die Bewertungsqualität des entwickelten Messinstruments wird anhand der Übereinstimmung der Ergebnisse der beiden unabhängigen Forscher beurteilt. Insgesamt wurden 23 % der Add-ons eindeutig codiert, bei 45 % betrug der Unterschied in der Bewertung eine Komplexitätsstufe und bei 32 % zwei Komplexitätsstufen. Somit können 68 % der Kodierungen als zuverlässig angesehen werden, da sie entweder übereinstimmend oder mit einer Abweichung von einer Stufe bewertet wurden. Die Übereinstimmung mit mehr als zwei Dritteln lässt auf die Verlässlichkeit des Kodierleitfadens schließen.

7 Diskussion

Die Einführung von LCP/NCP stellt für Unternehmen eine strategische Entscheidung dar, die sowohl Chancen als auch Herausforderungen mit sich bringt. Der vorliegende Beitrag verdeutlicht, dass die Eignung einer LCP/NCP nicht allein von den Grundfunktionen der Plattform abhängt, sondern maßgeblich auch von der Vielfalt und Komplexität der im zugehörigen Store verfügbaren Add-ons beeinflusst wird. Letztere spiegeln eine Abbildung der möglichen Individualisierung wider. Das entwickelte Instrument zur Messung der Komplexität eines LCP/NCP-Add-on-Stores stellt somit ein Instrument zur Bewertung und zum Vergleich alternativer Plattformen hinsichtlich ihrer Eignung zur Abdeckung spezifischer Geschäftsanforderungen dar.

7.1 Neuer Ansatz zur Messung des Komplexitätsgrades bei LCP/NCP

Dieser Beitrag stellt einen neuartigen Ansatz zur Messung der Komplexität von Add-ons in LCP/NCP Stores vor, der auf der Analyse textueller Beschreibungen basiert. Im Gegensatz zu traditionellen Methoden, die auf Softwaremetriken wie der Anzahl der Codezeilen (Rosenberg 1997), der Zählung von Operatoren und Operanden (Halstead 1977) oder der Analyse von Datenabrufen (Lake und Cook 1994). Diese sind nicht auf die Besonderheiten von LCP/NCP geeignet und zugeschnitten, da diese Plattformen stark auf visuelle und deklarative Techniken setzen (Fischer et al. 2021).

Der Messansatz wurde mit Blick auf die spezifischen Charakteristika von LCP/NCP entwickelt und berücksichtigt dabei deren Besonderheiten. Das zugrundeliegende Messinstrument basiert auf einem Kodierleitfaden, der aus der aktuellen Forschung zu LCP/NCP abgeleitet und durch Experteneinschätzungen validiert wurde. Der Leitfaden erlaubt es, ein differenziertes Verständnis der Komplexität von Add-ons zu gewinnen, wodurch eine wertvolle Unterstützung bei der Auswahl der geeigneten Plattform für spezifische Unternehmensanforderungen gewährleistet wird.

7.2 Entscheidungshilfe für Unternehmen

Die Messung der Komplexität eines Add-on Stores bietet eine neue Perspektive zur Bewertung der funktionalen Flexibilität und Anpassungsfähigkeit eines LCP/NCP. Unternehmen können dieses Instrument nutzen, um fundierte Entscheidungen darüber zu treffen, welche Plattform ihren spezifischen Anforderungen am besten gerecht werden könnte. Die Möglichkeit, auf ein breites und vielfältiges Angebot an Add-ons zuzugreifen, spielt eine wichtige Rolle bei der Nutzenbetrachtung einer LCP/NCP.

Hinsichtlich der Eignung der Technologie insgesamt erscheint das Einsatzprofil von LCP/NCP aufgrund des Konzeptes sowie der Umsetzung für einen bestimmten Komplexitätsbereich geeignet. Während einfache Anforderungen besser direkt im bestehenden Informationssystem umgesetzt werden, sind hochkomplexe Anforderungen aufgrund ihrer Anforderungen an die Gestaltungsfreiheit von LCP/NCP nicht geeignet, da diese die standardisierten Möglichkeiten von LCP/NCP überstei-

gen (Elshan et al. 2023). In der Praxis sind jedoch das Profil und die Funktionalitäten, die der LCP-Add-on Store in Form von Add-ons bietet, für Unternehmen entscheidend.

Die entwickelte Methodik stellt für Unternehmen einen Ansatz dar, welcher eine systematische Analyse der in den LCP-Add-on Stores angebotenen Funktionen von Plattformen ermöglicht. Der Ansatz erlaubt einen Vergleich verschiedener Plattformen im Rahmen der Technologieeinführung. Dadurch lassen sich differenzierte Erkenntnisse über die Vorteile und Einschränkungen einzelner LCP/NCP gewinnen. Somit werden Unternehmen befähigt, geeignete LCP/NCP auf Basis eines detaillierten Komplexitätsprofils zu identifizieren.

Dieser Beitrag konzentriert sich bei der Bewertung von Alternativen LCP/NCP darauf, die Eignung einer Plattform anhand der Add-ons in den LCP-Add-on Stores zu beurteilen. Dabei werden die unterschiedlichen Grundfunktionen der Plattformen nicht berücksichtigt, sondern es wird vereinfachend davon ausgegangen, dass sie als Lösungskategorie vergleichbare Funktionen bieten.

Des Weiteren beschränkt sich die Untersuchung auf öffentlich verfügbare Add-ons aus den LCP-Add-on Stores. Viele Unternehmen entwickeln und nutzen individuelle Lösungen, die nicht öffentlich zugänglich sind. Die Analyse schließt somit einen bedeutenden Teil von Unternehmensanwendungen aus, was die Generalisierbarkeit der Ergebnisse einschränkt. Ebenfalls im Hinblick auf die Repräsentativität ist zu berücksichtigen, dass bisher keine NCP Add-on Stores in die Untersuchung einbezogen wurden.

Eine weitere Limitation ergibt sich aus der erstmaligen manuellen Klassifikation der Add-Ons hinsichtlich ihrer Komplexität. Diese Klassifikation unterliegt einer gewissen Subjektivität, da die Beurteilung der Komplexität von individuellen Einschätzungen abhängig ist. Trotz der Bemühungen, einen einheitlichen Kodierleitfaden anzuwenden, können individuelle Interpretationen zu Abweichungen in der Klassifizierung führen. Zusätzlich wurde die Evaluierung des Kodierleitfadens bislang hauptsächlich auf die Intercoder-Reliabilität fokussiert, ohne eine externe Validierung der tatsächlichen Komplexität der bewerteten Add-ons vorzunehmen.

7.3 Zukünftige Forschung

Der bisherige manuelle Ansatz mittels Kodierleitfaden könnte in einen automatisierten Ansatz überführt werden. Dabei könnten Funktionen aus den Beschreibungen über NLP-Techniken abgeleitet und analysiert werden.

Eine breitere Datenanalyse, die über die bisher einbezogenen LCP-Add-on Stores hinausgeht, würde die Repräsentativität der Analyse verbessern. Denkbar wäre auch eine differenziertere Betrachtung der Komplexität der Add-ons. Anstelle einer eindimensionalen Betrachtung sollten mehrere Dimensionen der Komplexität berücksichtigt werden, um ein differenzierteres Verständnis der Herausforderungen und Anforderungen bei der Entwicklung von Add-Ons auf LCP/NCP zu erhalten. Ein mehrdimensionales Komplexitätsmodell kann verschiedene Aspekte wie technische Schwierigkeit, erforderliches Fachwissen und Integrationsaufwand berücksichtigen. Abhängigkeiten zwischen den Dimensionen können ebenfalls berücksichtigt werden.

Automatisierte Ansätze für die Analyse, wie zum Beispiel die Anwendung von Topic Modelling, bieten das Potenzial, die Effizienz und Objektivität der Datenanalyse zu erhöhen. Durch den Einsatz solcher Methoden könnten breitere Analysen durchgeführt werden, die nicht nur die Komplexität, sondern auch weitere Aspekte erfassen.

8 Schlussfolgerung

Die vorliegende Studie leistet einen Beitrag zum Verständnis der Potenziale von LCP und NCP. Dazu wird ein Messinstrument zur Bewertung der Komplexität von Add-ons vorgestellt. Das Instrument basiert auf einer textuellen Beschreibung der Add-ons, welche mittels eines spezifischen Kodierleitfadens analysiert werden. Dies erlaubt eine charakteristische Evaluierung der durch Add-ons innerhalb der Plattformen bereitgestellten Funktionen.

Die Anwendung des Messinstruments ermöglicht es Unternehmen, im Vorfeld der Technologieadoption fundierte Entscheidungen für oder gegen eine spezifische Plattform zu treffen. Sie sind in der Lage, zu erkennen, welche alternativen LCP/NCP für ihre spezifischen Anwendungsanforderungen mittels verfügbarer Add-ons abgedeckt werden könnten. Die Ergebnisse dieser Studie bieten somit eine Grundlage für strategische Überlegungen von Unternehmen, die den Einsatz von LCP/NCP vorsehen.

Tab. 6 Abgeleitete Anforderungen basierend auf der LCP/NCP-Literatur

Literatur	Abgeleitete Anforderung an Nutzer	Schwierigkeitsgrad aus Literatur	Geschätzte Komplexitäts-ausprägung	Begründung
Bies et al. (2022); Hollick (2021); Park et al. (2022)	Erstellen von Dashboards	Einfach	1	Grundfunktionen LCP/NCP
Bies et al. (2022); Pinho et al. (2023)	Erstellen von Formularen	Einfach	1	Grundfunktionen LCP/NCP
Alamin et al. (2021, 2023)	Erstellen von dynamischen Formularen	Einfach	2	Grundfunktionen LCP/NCP
Alamin et al. (2023)	Erstellung interaktiver Reports	Einfach	2	Grundfunktionen LCP/NCP
Bernsteiner et al. (2022)	Anbindung diverser Daten-banktabellen	Schwer	2	Mittels Grundfunktionen der LCP/NCP ist häufig eine Anbindung mit Login der Datenbanken einfach möglich
Bernsteiner et al. (2022)	Datentransfer zwischen un-terschiedlichen Applikationen ermöglichen	Schwer	4	Hohe Kenntnisse notwendig (u. a. für Container-Lösungen)
Alamin et al., (2021, 2023); Bernsteiner et al. (2022)	Anbindung an APIs	Schwer	4	Hohe Kenntnisse in Webprogrammierung not-wendig (u. a. HTTP und REST-Befehle)
Alamin et al. (2023); Binzer und Winkler (2023); Hollick (2021); Käss et al. (2022)	Integration zu anderen Syste-men ermöglichen	Schwer	4	Hohe Kenntnisse notwendig (u. a. Authentifizie-rungsmethoden, Schnittstellenanpassung)
Alamin et al. (2021, 2023)	Erstellung einer CRUD-Appli-kation	Einfach	2	Grundfunktionen LCP/NCP (automatisches ein-lesen der Datenbank, Daten ergänzen oder ändern sowie löschen)
Alamin et al. (2023)	Implementierung von Muster-erkennung	Einfach	2	Grundfunktionen LCP/NCP (abhängig von der Lösung)
Alamin et al. (2023)	Durchführung von Stilmanipu-lation	Einfach	1	Grundfunktionen LCP/NCP

Tab. 6 (Fortsetzung)

Literatur	Abgeleitete Anforderung an Nutzer	Schwierigkeitsgrad aus Literatur	Geschätzte Komplexitätsausprägung	Begründung
Alamin et al. (2023)	Bearbeitung von Daten und Zeit	Einfach	2	Grundfunktionen LCP/NCP
Alamin et al. (2023)	Formatierung von Daten zum Einlesen	Einfach	2	Grundfunktionen LCP/NCP. Fähigkeiten zum Verständnis des Aufbaus von Daten notwendig
Alamin et al. (2021, 2023)	Verwaltung von Entitätsbeziehungen	Einfach	2	Grundfunktionen LCP/NCP
Alamin et al. (2023)	Anpassung der SQL-Syntax	Einfach	3	Erfahrungen in SQL notwendig, um Syntax zu verstehen
Alamin et al. (2023)	Datenfilterung aus Datenbanken	Einfach	2	Grundfunktionen LCP/NCP über z. B. Funktionsabfragen
Alamin et al. (2023)	Aktualisierung und Kompatibilitätsprüfung	Einfach	2	Überprüfung der Versionsnummer im LCP/NCP Store
Alamin et al. (2023)	Bereitstellung der Datensicherheit und Replikation	Einfach	3	Mittels Anleitungen möglich. Je nach LCP/NCP abhängig und komplexer
Alamin et al. (2023)	Erstellung von dynamischen Datenfiltern	Einfach	2	Grundfunktionen LCP/NCP
Alamin et al. (2023)	Dynamische Datenbindung	Einfach	2	Grundfunktionen LCP/NCP
Alamin et al. (2023)	Erstellung von Diagrammen und Grafiken	Einfach	1	Grundfunktionen LCP/NCP
Alamin et al. (2023)	Erstellung von Manipulationen in Dialogen	Einfach	2	Grundfunktionen LCP/NCP duech bearbeiten oder entfernen von Benutzerinteraktion, Formularfeldern und anderen UI-Elementen
Alamin et al. (2023)	Verarbeitung von E-Mails	Einfach	2	Grundfunktionen LCP/NCP oder Konnektoren zu E-Mail Clients vorhanden

Tab. 6 (Fortsetzung)

Literatur	Abgeleitete Anforderung an Nutzer	Schwierigkeitsgrad aus Literatur	Geschätzte Komplexitätsausprägung	Begründung
Alamin et al. (2021, 2023)	Dynamische Ereignisverarbeitung	Schwer	3	Grundfunktionen LCP/NCP. Abhängig von der Komplexität der dynamischen Ereignisse
Alamin et al. (2023)	Konfigurationsmanagement	Einfach	2	Grundfunktionen LCP/NCP, ggf. Änderungen in Config-Dateien über Anleitungen möglich
Alamin et al. (2023)	Ermittlung und Darstellung von Bedingungen aus BPMN	Einfach	2	Grundfunktionen LCP/NCP
Alamin et al. (2023); Hintsch et al. (2021)	Durchführung von Tests	Einfach	2	Durchführung von Usability Tests, um alle Masken auf funktionsfähigkeit zu überprüfen
Alamin et al. (2023)	Migration und Setup von Datenbanken	Einfach	2	Grundfunktionen LCP/NCP über Migrationstools
Alamin et al. (2023)	Erstellung von Datenbank gespeicherten Prozeduren	Einfach	2	Grundfunktionen LCP/NCP
Alamin et al. (2023)	Ausführung von plattformspezifischen Abfragen	Einfach	2	Grundfunktionen LCP/NCP
Alamin et al. (2023); Hollick (2021)	Verwaltung von Benutzerrollen	Schwer	3	Grundfunktionen LCP/NCP. Abhängig von der jeweiligen Verwaltungsoberfläche
Alamin et al. (2023)	Erstellung von dynamischen Seitenlayouts	Schwer	3	Grundfunktionen LCP/NCP jedoch werden nach Anforderungen Webkenntnisse (HTML, CSS) notwendig
Alamin et al. (2023)	Ausführung von Batch-Aufträgen im asynchronen Modus	Schwer	4	Es sind Programmierkenntnisse wie Python oder Javascript notwendig. In Teilen gibt es Grundfunktionen von LCP/NCP
Alamin et al. (2023)	Dateiverwaltung	Schwer	4	Erfordert in Teilen Programmierkenntnisse zur Speicherung und Umwandlung von Daten, Transformation vom Ursprungs- zum Zielformat sowie Handhabung von Bilddateien
Alamin et al. (2023)	Erstellung digitaler Signaturen	Schwer	4	Keine Grundfunktionen für LCP/NCP vorhanden. Es sind Programmierkenntnisse oder Paketkenntnisse notwendig

Tab. 6 (Fortsetzung)

Literatur	Abgeleitete Anforderung an Nutzer	Schwierigkeitsgrad aus Literatur	Geschätzte Komplexitätsausprägung	Begründung
Alamin et al. (2023)	Konfiguration des Hostings und Suchmaschinen-optimierung	Schwer	3	LCP/NCP bieten oft integrierte Funktionen und Tools zur Konfiguration des Hostings, wie z. B. die Bereitstellung auf Cloud-Plattformen oder das Konfigurieren von Servern und Datenbanken. Diese Tools sind in der Regel benutzerfreundlich gestaltet und ermöglichen es auch Nutzern mit wenig technischem Hintergrund, ihre Anwendungen erfolgreich zu hosten. Suchmaschinenoptimierung erfordert ein Verständnis in die Thematik, kann häufig über Grundfunktionen LCP/NCP eingebunden werden
Alamin et al. (2023)	Bereitstellung der Applikation	Schwer	4	DevOps-Praktiken sind häufig mit spezialisierten Aufwand bei der Bereitstellung von Applikationen verbunden. LCP/NCP bieten Grundfunktionen zur Bereitstellung. Bei komplexeren Bereitstellungsoptionen ist Fachwissen erforderlich
Alamin et al. (2021, 2023)	Verarbeitung von externen Webanfragen	Schwer	4	Erfordert Kenntnisse in der Abfrage (u.a. HTTP) sowie bei Protokollen (u. a. SOAP, REST)
Alamin et al. (2021, 2023)	Implementierung von Authentifizierung und Berechtigungen	Schwer	4	Wissen in OAuth oder benutzerdefinierten Rolle- und Zugriffsmechanismen ist notwendig
Alamin et al. (2023)	Verwaltung der Webservice Kommunikation	Schwer	4	Die Verwaltung der Webservice-Kommunikation umfasst das Senden, Empfangen und Verarbeiten von Daten über Webservices wie RESTful APIs oder SOAP. Diese Aufgabe erfordert ein Verständnis von Webtechnologien, Datenformate, Fehlerbehandlung und Netzwerkprotokollen

Tab. 6 (Fortsetzung)

Literatur	Abgeleitete Anforderung an Nutzer	Schwierigkeitsgrad aus Literatur	Geschätzte Komplexitätsausprägung	Begründung
Alamin et al. (2023)	Verwaltung von Bibliotheksabhängigkeiten	Schwer	3	LCP/NCP verfügen in Teilen über Möglichkeiten, zusätzliche Programmbibliotheken einzubinden. Mittels Tutorials aus den LCP/NCP kann eine Anbindung möglich sein
Alamin et al. (2023)	Verwaltung von Nachrichtenwarteschlangen	Schwer	4	Diese Aufgabe erfordert ein Verständnis für Messaging-Protokolle, Warteschlangenmechanismen sowie Fehlerbehandlung, was ohne Programmierkenntnisse schwierig ist
Alamin et al. (2021)	Konfiguration von Cloud und On-Premises-Umgebungen	Schwer	4	Die Konfiguration von Cloud- und On-Premises-Umgebungen erfordert ein tiefes Verständnis von Cloud-Infrastrukturen, Netzwerken, Sicherheit, Skalierung und anderen komplexen Konzepten
Alamin et al. (2021); Hollick (2021)	Anpassung der Benutzeroberfläche	Einfach	2	Grundfunktionen LCP/NCP oder Anpassungen mittels Tutorials über CSS
Alamin et al. (2021)	Verwaltung des Datenspeichers	Einfach	2	Grundfunktionen LCP/NCP bzw. Integrationsmöglichkeiten über verfügbare Konnektoren
Hintsch et al. (2021)	Verwaltung der Sicherheit	Schwer	4	Die Verwaltung der Sicherheit ist eine komplexe Aufgabe, die ein tiefes Verständnis von Sicherheitskonzepten, Best Practices und ISO-Normen beinhaltet

Tab. 6 (Fortsetzung)

Literatur	Abgeleitete Anforderung an Nutzer	Schwierigkeitsgrad aus Literatur	Geschätzte Komplexitätsausprägung	Begründung
Hollick (2021)	Anpassung von Import- und Exportfunktionen für die Bereitstellung im gesamten Unternehmen	Einfach	2	Grundfunktionen LCP/NCP über Teammodul
Hollick (2021)	Erstellung von benutzerdefinierten Datenfeldern für spezifische Abfragen und Berichterstattung	Einfach	1	Grundfunktionen LCP/NCP
Bies et al. (2022)	Entwicklung von Ressourcenmanagement	Einfach	3	Über vordefinierte Module und Komponenten kann ein Ressourcenmanagement erstellt werden
Bies et al. (2022)	Entwicklung von Raumbuchungstools	Einfach	3	Über vordefinierte Module und Komponenten kann ein Raumbuchungstool erstellt werden
Bies et al. (2022)	Entwicklung eines Inventarisierungsprogramms	Einfach	3	Über vordefinierte Module und Komponenten kann ein Inventarisierungsprogramms erstellt werden
Experten	Müssen in der Lage sein, bestimmte Programmiersprachen und Skriptsprachen (Java, Python, .net, Javascript)	Schwer	5	Große Kenntnisse in Programmierung notwendig
Experten	Müssen in der Lage sein, bestimmte Frameworks und Technologien einzusetzen (u. a. Angular, REACT, SOAP, Blockchain)	Schwer	5	Große Kenntnisse in Programmierung notwendig

Tab. 7 Beispielbeschreibungen nach Komplexitätskategorien mit Begründung der Einordnung

Gesamtkomplexität	Beschreibung des Add-ons mit Begründung zur Einordnung
Sehr geringe Komplexität (1)	This is a **commission app** which is going to **calculate** commission for the sales team. 1. There are few types of commission eg: **Basic, Revenue, Product Base, Product Revenue** and **Tiered commission**. we can select he commission on our need. 2. Create a Goal and then select the commission type. 3. Then on the basis of commission type you can easily calculate your generated commission. **Begründung der Einordnung:** – *Das Add-on bietet grundlegende Funktionen wie die Auswahl verschiedener und die Erstellung eines Vertriebsziels.* – *Diese Funktionen sind in einem abgegrenzten Bereich und haben einen klar definierten Zweck, nämlich die Berechnung von Provisionen. Die Auswahl des Provisionstyps und die anschließende Berechnung basieren auf einfachen logischen Operationen, die mit den Grundfunktionen der LCP/NCP leicht abbildbar sind.* – *Es sind keine komplexen Verzweigungen oder verschachtelten Datenbankabfragen erforderlich, sondern eher einfache Berechnungen und logische Zuordnungen*
Geringe Komplexität (2)	The fewer people in a room, the better—at least from the point of view of corona prevention. This is the idea behind our **room booking app**. Employees can use the app to book **unused rooms**, e.g. conference rooms or canteens. A **maximum capacity** can be defined for each room, which allows for safer working. Once the maximum capacity has been reached, the room can no longer be booked. This allows employees to be distributed better throughout the building, and because rooms can also be booked several days in advance, employees can assess the occupancy of the building before they go to the office—and stay at home if in doubt. Employees need an Office 365 license to use the app. The room booking app works with current Android and Apple devices as well as current web browsers. Architecture: There are two versions of the app. One for the **end user**, who can only book rooms. The other version is for the **administrator**, who can also create and manage rooms. They can also see who booked which room and when. **Begründung der Einordnung:** – *Es handelt sich um eine CRUD-Applikation die die Buchung, Verfügbarkeit und Verwaltung von Räumen umfasst. Es gibt unterschiedliche Zugriffsrechte für Endbenutzer und Administratoren, wobei Administratoren zusätzliche Funktionen zur Raumverwaltung und Einsicht in Buchungen haben.* – *Die Anwendung umfasst mehrere Verzweigungen und logische Bedingungen, insbesondere bei der Überprüfung der Raumverfügbarkeit und der Einhaltung der maximalen Kapazität.* – *Es sind logische Operatoren und gegebenenfalls verschaltete Operatoren bei der Überprüfung der Raumverfügbarkeit notwendig*

Tab. 7 (Fortsetzung)

Gesamtkomplexität	Beschreibung des Add-ons mit Begründung zur Einordnung
Mittlere Komplexität (3)	Are you tired of drowning in paperwork and inefficiencies when it comes to managing expenses? Introducing our game-changing solution, Intech's **Expense Management**! Say goodbye to the hassle of manual data entry, slow approval processes, and the stress associated with expense reporting. With Expense Management, you gain the power to **submit, track, and manage** expenses seamlessly, all at your fingertips. Why Choose Expense Management? Accessible Anytime, Anywhere: Submit expenses from any device, wherever you are. – *Web-Based, User-Friendly Interface: Experience a smooth, intuitive platform for effortless expense management.* – *Real-Time Notifications: Stay in the loop with status updates directly in Dynamics 365 and via email.* – *One-Click Approvals: Speed up the approval process with a simple click, enhancing efficiency.* – *Seamless Integration: Connect effortlessly with your existing solutions, including Dynamics 365.* – *Data Entry Made Easy: Minimize errors with zero to less manual data entry.* – *Strong Control and Validation: Ensure accuracy through robust technical validation for any discrepancies.* Ready to revolutionize your expense management experience? Try Expense Management now and streamline your financial processes today! Check out our brochure to learn more. **Begründung der Einordnung:** – *Es handelt sich um eine komplexe CRUD-Anwendung, die die Einreichung, Verfolgung und Verwaltung von Ausgaben umfasst.* – *Die Anwendung bietet mehrere Funktionen wie Echtzeit-Benachrichtigungen, Ein-Klick-Genehmigungen und nahtlose Integration mit bestehenden Systemen*

Tab. 7 (Fortsetzung)

Gesamtkomplexität	Beschreibung des Add-ons mit Begründung zur Einordnung
Hohe Komplexität (4)	Can you control your organization's day-to-day risks? Can you predict the outcomes of these risks to your organization? If your answer is no, we can help you with our "Risk management" system. 1. This system is developed in accordance with the **ISO 31000** standard and is capable of registering, assessing, responding, monitoring, and reporting on all levels of risk with the organizations. 2. Key features: **Register risk by category and divisions**. Two level assessment for more real score. Each registered risk including consequences, mitigation plan. Monitoring risks until it is mitigated. You can track the processes of the risks in Realtime. Every data that created from the system can be displayed on the live dashboards to help you analyze your risk or lower the decision-making process. 3. What value "Risk management" system give you. Reduce time for register, assessing and planning risk. Monitoring assessment and mitigation plan for anywhere and anytime. Increasing collaboration and communication. More controlled, reducing human error. **Begründung der Einordnung:** – *Das System ist umfassend und deckt alle Aspekte des Risikomanagements gemäß ISO 31000 ab, einschließlich der Registrierung, Bewertung, Überwachung und Berichterstattung. Es bietet eine zweistufige Risikobewertung und erfordert eine detaillierte Erfassung von Konsequenzen und Minderungsplänen.* – *Die kontinuierliche Überwachung und Echtzeitverfolgung der Risikoprozesse erfordert komplexe Logiken und Integrationen. Die Implementierung von Live-Dashboards und Echtzeitdatenanzeige zur* Unterstützung der Entscheidungsfindung
Sehr hohe Komplexität (5)	Data Mask tool for Dataverse is a powerful tool designed to **secure your sandbox environments by replacing sensitive data with dummy data**. The tool employs techniques such as **masking, anonymization**, and **obfuscation** to ensure that confidential data is safeguarded during development, testing, and training activities. With Data Mask tool, users can effortlessly select the fields to be masked or deleted, as well as employ pre-built formats for generating realistic yet fictitious data. You can use the Test Drive option to give the tool a spin. The tool is offered as a model-driven app. As part of the configuration, you must specify which tables/attributes to mask. Please contact us; we would be pleased to provide a demo. **Begründung der Einordnung:** – *Das Add-on verwendet fortgeschrittene Techniken zur Datenmaskierung, Anonymisierung und Verschleierung, die über einfache logische Operationen hinausgehen.* – *Die Konfiguration und Anwendung von vorgefertigten Datenformaten erfordert ein tiefes Verständnis der Datenstruktur und der Anforderungen an die Datensicherheit*

Tab. 8 Verteilung der Kategorien nach Gesamtkomplexität aus den MS Power Apps Store

Kategorie	Sehr geringe Gesamtkomplexität		Geringe Gesamtkomplexität		Mittlere Gesamtkomplexität		Hohe Gesamtkomplexität		Sehr hohe Gesamtkomplexität		Gesamt
		(%)		(%)		(%)		(%)		(%)	
Sales	3	1,3	69	30	25	10,9	90	39,1	43	18,7	*230*
Customer Service	3	1,9	51	32,5	14	8,9	58	36,9	31	19,7	*157*
IT & Management Tools	8	5,8	46	33,1	19	13,7	44	31,7	22	15,8	*139*
Human Resources	15	14,3	63	60,0	9	8,6	15	14,3	3	2,9	*105*
Productivity	7	7,9	31	34,8	5	5,6	28	31,5	18	20,2	*89*
Collaboration	8	10,4	39	50,6	7	9,1	19	24,7	4	5,2	*77*
Operations & Supply Chain	3	3,9	30	39,5	6	7,9	27	35,5	10	13,2	*76*
Finance	2	3,6	18	32,7	5	9,1	14	25,5	16	29,1	*55*
Compliance & Legal	1	2,2	8	17,4	2	4,3	22	47,8	13	28,3	*46*
Marketing	1	2,6	13	34,2	1	2,6	12	31,6	11	28,9	*38*
Analytics	1	2,9	12	35,3	3	8,8	14	41,2	4	11,8	*34*
Sonstige	0	0	9	23,7	1	2,6	15	39,5	13	34,2	*38*
Gesamt	*53*	–	*391*	–	*95*	–	*358*	–	*187*	–	*1084*

Funding Open Access funding enabled and organized by Projekt DEAL.

Literatur

Adrian B, Hinrichsen S, Nikolenko A (2020) App development via low-code programming as part of modern industrial engineering education. In: Nunes IL (Hrsg) Advances in human factors and systems interaction. Springer, S 45–51. https://doi.org/10.1007/978-3-030-51369-6_7

Alamin MAA, Malakar S, Udin S, Afroz S, Haider BT, Iqbal A (2021) An Empirical Study of Developer Discussions on Low-Code Software Development Challenges. IEEE/ACM 18th International Conference on Mining Software Repositories (MSR). International Conference on Mining Software Repositories (MSR). März, Bd. 21. https://doi.org/10.1109/MSR52588.2021.00018

Alamin MAA, Uddin G, Malakar S, Afroz S, Haider T, Iqbal A (2023) Developer discussion topics on the adoption and barriers of low code software development platforms. EMPIRICAL SOFTWARE ENGINEERING, 28(1), Article 1. https://doi.org/10.1007/s10664-022-10244-0

Almutairi A, Naeem MA, Weber G (2022) Understanding enterprise systems adaptability: An exploratory survey. Procedia Comput Sci 197:743–750. https://doi.org/10.1016/j.procs.2021.12.196

Balint B (2017) Maximizing the value of packaged software customization: a nonlinear model and simulation. Int J Enterp Inf Sys 13(1):1–16. https://doi.org/10.4018/ijeis.2017010101

Basili VR (1988) Models and metrics for software management and engineering https://ntrs.nasa.gov/citations/19880014816

Becker F, Meyer M, Redlich B, Siemon D, Lattemann C (2020) Open KMU: Mit Action Design Research und Design Thinking gemeinsam innovieren. HMD 57(2):274–284 https://doi.org/10.1365/s40702-020-00604-z

Bender B (2021) The impact of integration on application success and customer satisfaction in mobile device platforms. In: Bender B (Hrsg) Platform coring on digital software platforms. Springer, S 79–118. https://doi.org/10.1007/978-3-658-34799-4_5

Bernsteiner R, Schlögl S, Ploder C, Dilger T, Brecher F (2022) Citizen vs professional developers differences and similarities of skills and training requirements for low code development platforms. In: ICERI2022 Proceedings, S 4257–4264. https://doi.org/10.21125/iceri.2022.1036

Bies L, Weber M, Greff T, Werth D (2022) A mixed-methods study of low-code development platforms: drivers of digital innovation in SME. 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), S 1–6. https://doi.org/10.1109/ICECCME55909.2022.9987920

Binzer B, Winkler TJ (2023) Low-coders, no-coders, and citizen developers in demand: examining knowledge, skills, and abilities through a job market analysis. Wirtschaftsinformatik 2023 Proceedings, Paderborn

Bock AC, Frank U (2021) Low-code platform. Bus Inf Syst Eng 63(6):733–740. https://doi.org/10.1007/s12599-021-00726-8

Boudreau K (2010) Open platform strategies and innovation: granting access vs. devolving control. Manag Sci 56(10):1849–1872. https://doi.org/10.1287/mnsc.1100.1215

Bratincevic J The low-code market could approach $ 50 billion by 2028. Forrester. https://www.forrester.com/blogs/the-low-code-market-could-approach-50-billion-by-2028/ (Erstellt: 29. Jan. 2024)

Brünninghaus M, Röcker C (2022) Low-code development in worker assistance systems: improving flexibility and adaptability. In: 2022 IEEE 20th International Conference on Industrial Informatics (INDIN), S 366–373. https://doi.org/10.1109/INDIN51773.2022.9976178

Cabot J (2020) Positioning of the low-code movement within the field of model-driven engineering. In: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, S 1–3. https://doi.org/10.1145/3417990.3420210

Cai FZ, Huang SY, Kessler TS, Fottner FJ (2022) A case study: digitalization of business processes of SMEs with low-code method. IFAC-PapersOnLine 55(10):1840-1845. https://doi.org/10.1016/j.ifacol.2022.09.666

Cennamo C, Santalo J (2013) Platform competition: strategic trade-offs in platform markets. Strat Mgmt J 34(11):1331–1350. https://doi.org/10.1002/smj.2066

Cho ES, Kim MS, Kim SD (2001) Component metrics to measure component quality. In: Proceedings Eighth Asia-Pacific Software Engineering Conference, S 419–426. https://doi.org/10.1109/APSEC.2001.991509

Curty, S., Härer, F., Fill, HG (2023) Design of blockchain-based applications using model-driven engineering and low-code/no-code platforms: a structured literature review. Softw Syst Model 22:1857–1895. https://doi.org/10.1007/s10270-023-01109-1

Davison RM, Wong LHM, Ou CXJ, Alter S (2021) The coordination of workarounds: insights from responses to misfits between local realities and a mandated global enterprise system. Info Manag 58(8):103530. https://doi.org/10.1016/j.im.2021.103530

Deep L (2024) DeepL translate: the world's most accurate translator. https://www.deepl.com/translator

Elshan E, Dickhaut E, Ebel P (2023) An investigation of why low code platforms provide answers and new challenges. In: Proceedings of 56th International Conference on System Sciences (HICSS). Mauii, Hawaii

Feldman R, Sanger J (2007) The text mining handbook: advanced approaches in analyzing unstructured data, 1. Aufl. University Press, Cambridge. https://doi.org/10.1017/CBO9780511546914

Fischer J, Vogel-Heuser B, Schneider H, Langer N, Felger M, Bengel M (2021) Measuring the overall complexity of graphical and textual IEC 61131-3 control software. IEEE Robotics Autom Lett 6(3):5784–5791. https://doi.org/10.1109/LRA.2021.3084886

Fryling M (2010) Estimating the impact of enterprise resource planning project management decisions on post-implementation maintenance costs: A case study using simulation modelling. Enterp Inf Sys 4(4):391–421. https://doi.org/10.1080/17517575.2010.519785

Gartner (2022) Gartner forecasts worldwide low-code development technologies market to grow 20% in 2023. Gartner. https://www.gartner.com/en/newsroom/press-releases/2022-12-13-gartner-forecasts-worldwide-low-code-development-technologies-market-to-grow-20-percent-in-2023 (Erstellt: 13.12.2022)

Ghazawneh A, Henfridsson O (2015) A paradigmatic analysis of digital application marketplaces. J Inf Technol 30(3):198–208. https://doi.org/10.1057/jit.2015.16

Haddara M, Gøthesen S, Langseth M (2022) Challenges of cloud-ERP adoptions in SMEs. Procedia Comput Sci 196:973–981. https://doi.org/10.1016/j.procs.2021.12.099

Haile N, Altmann J (2016) Structural analysis of value creation in software service platforms. Electron Mark 26(2):129–142. https://doi.org/10.1007/s12525-015-0208-8

Halstead MH (1977) Elements of Software Science. Computer Science library, Vol. 7. Elsevier, New York, NY

Hintsch J, Staegemann D, Volk M, Turowski K (2021) Low-code development platform usage: towards bringing citizen development and enterprise IT into harmony. ACIS 2021 Proceedings, 11. https://aisel.aisnet.org/acis2021/11

Hirzel M (2023) Low-code programming models. Communications of the ACM 66(10):76–85. https://doi.org/10.1145/3587691

Hitt LM, Wu DJ, Zhou X (2002) Investment in enterprise resource planning: business impact and productivity measures. J Manag Inf Syst 19(1):71–98. https://doi.org/10.1080/07421222.2002.11045716

Hollick R (2021) How low-code / no-code can be used in ERP | ERP. SYSPRO Blog. https://www.syspro.com/blog/owning-or-running-erp/how-low-code-no-code-can-be-used-in-erp/ (Erstellt: 02.09.2021)

Karahanna E, Straub DW, Chervany NL (1999) Information technology adoption across time: a cross-sectional comparison of pre-adoption and post-adoption beliefs. MIS Q 23(2):183–213. https://doi.org/10.2307/249751

Käss S, Strahringer S, Westner M (2022) Drivers and inhibitors of low code development platform adoption. 2022 IEEE 24th Conference on Business Informatics (CBI). Bd. 1, S 196–205. https://doi.org/10.1109/CBI54897.2022.00028
Käss S, Strahringer S, Westner M (2023) Practitioners' perceptions on the adoption of low code development platforms. IEEE Access 11:29009–29034. https://doi.org/10.1109/ACCESS.2023.3258539
Lake A, Cook C (1994) Use of factor analysis to develop OOP software complexity metrics. Proc. 6th annual oregon workshop on software metrics, Silver falls, Oregon.
Lethbridge TC (2021) Low-code is often high-code, so we must design low-code platforms to enable proper software engineering. In: Margaria T, Steffen B (Hrsg) Leveraging applications of formal methods, verification and validation. ISoLA 2021. Lecture Notes in Computer Science, Bd. 13036. Springer, Cham. https://doi.org/10.1007/978-3-030-89159-6_14
Lokuge S, Sedera D (2017) Turning dust to gold: How to increase inimitability of enterprise system. PACIS 2017 Proceedings, 228. https://aisel.aisnet.org/pacis2017/228
Lourenço M, Gasiba TE, Pinto-Albuquerque M (2023) You are doing it wrong—on vulnerabilities in low code development platforms. CYBER 2023: The Eighth International Conference on Cyber-Technologies and Cyber-Systems, S 12–18
MacKenzie, Podsakoff (2011) Construct measurement and validation procedures in MIS and behavioral research: integrating new and existing techniques. MIS Q 35(2):293–334. https://doi.org/10.2307/23044045
McCabe TJ (1976) A complexity measure. IEEE Trans Softw Eng 2(4):308–320. https://doi.org/10.1109/TSE.1976.233837
Metrôlho J, Araújo R, Ribeiro F, Castela N (2019) An approach using a low-code platform for retraining professionals to ICT. EDULEARN19 Proceedings, S 7200–7207. https://doi.org/10.21125/edulearn.2019.1719
Microsoft (2024) Appsource—business Apps. https://appsource.microsoft.com/de-de/marketplace/apps?product=powerapps
Müller RM, Kijl B, Martens JKJ (2011) A comparison of inter-organizational business models of mobile app stores: there is more than open vs. closed. J Theor Appl Electron Commer Res 6(2):63–76. https://doi.org/10.4067/S0718-18762011000200007
Nikou S, Bouwman H, De Reuver M (2014) A consumer perspective on mobile service platforms: a conjoint analysis approach. Commun Assoc Inf Syst. https://doi.org/10.17705/1CAIS.03482
Offermann P, Levina O, Schönherr M, Bub U (2009) Outline of a design science research process. Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology—DESRIST '09, S 1–11. https://doi.org/10.1145/1555619.1555629
Olleros X (2008) The lean core in digital platforms. Technovation 28(5):266–276. https://doi.org/10.1016/j.technovation.2007.07.004
Ono Y, Saito D, Washizaki H, Fukazawa Y (2024) Measuring complexity in visual programming for elementary school students. J Inf Process 32(0):103–112. https://doi.org/10.2197/ipsjjip.32.103
Outsystems (2024) Search forge: assets—outsystems. https://www.outsystems.com/forge/
Park K, Mott B, Lee S, Gupta A, Jantaraweragul K, Glazewski K, Scribner JA, Ottenbreit-Leftwich A, Hmelo-Silver CE, Lester J (2022) Investigating a visual interface for elementary students to formulate AI planning tasks. J Comput Lang 73:101157. https://doi.org/10.1016/j.cola.2022.101157
Parthasarathy S, Sharma S (2017) Impact of customization over software quality in ERP projects: an empirical study. Softw Qual J 25(2):581–598. https://doi.org/10.1007/s11219-016-9314-x
Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S (2007) A design science research methodology for information systems research. J Manag Info Sys 24(3):45–77. https://doi.org/10.2753/MIS0742-1222240302
Pega (2024) Pega marketplace search | Pega community. https://community.pega.com/marketplace/search
Picek R (2023) Low-code/no-code platforms and modern ERP systems. In: 2023 International Conference on Information Management (ICIM), S 44–49 https://doi.org/10.1109/ICIM58774.2023.00014
Pinho D, Aguiar A, Amaral V (2023) What about the usability in low-code platforms? A systematic literature review. J Comput Lang 74:101185. https://doi.org/10.1016/j.cola.2022.101185
Rajagopal P (2002) An innovation—Diffusion view of implementation of enterprise resource planning (ERP) systems and development of a research model. Info Manag 40(2):87–114. https://doi.org/10.1016/S0378-7206(01)00135-5
Rokis K, Kirikova M (2023) Exploring low-code development: a comprehensive literature review. Complex Syst Inform Model Q (CSIMQ) 36:68–86. https://doi.org/10.7250/csimq.2023-36.04
Rosenberg J (1997) Some misconceptions about lines of code. In: Proceedings fourth international software metrics symposium, S 137–142. https://doi.org/10.1109/METRIC.1997.637174

Ruan LL, Patton EW, Tissenbaum M (2017) Evaluations of programming complexity in app inventor. In: Conference Proceedings of International Conference on Computational Thinking Education 2017, S 2–5

Rymer JR (2017) Vendor landscape: a fork in the road for low-code development platforms. Analyst report. Forrester Research

Sahay A, Indamutsa A, Di Ruscio D, Pierantonio A (2020) Supporting the understanding and comparison of low-code development platforms. 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), S 171–178. https://doi.org/10.1109/SEAA51224.2020.00036

Schneider GM, Sedlmeyer RL, Kearney J (1981) On the complexity of measuring software complexity. Proceedings of the May 4–7, 1981, National Computer Conference on—AFIPS '81, S 317–322. https://doi.org/10.1145/1500412.1500456

scrapy (2023) Scrapy | A fast and powerful scraping and web crawling framework (version 2.11.0) [python]. Zyte group limited. https://scrapy.org/

Song C, Park K, Kim BC (2013) Impact of online reviews on mobile app sales: open versus closed platform comparison. PACIS 2013 Proceedings, 12. https://aisel.aisnet.org/pacis2013/12

Sonnleithner L, Zoitl A (2020) A software measure for IEC 61499 basic function blocks. 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). Bd. 1, S 997–1000

Teece D, Peteraf M, Leih S (2016) Dynamic capabilities and organizational agility: risk, uncertainty, and strategy in the innovation economy. Calif Manage Rev 58(4):13–35. https://doi.org/10.1525/cmr.2016.58.4.13

Totterdale RL (2018) Case study the utilization of low code development technology to support research data collection. Issues in Inform Syst 19(2). https://doi.org/10.48009/2_iis_2018_132-139

Wang S, Wang H (2021) A teaching module of no-code business app development. J Inform Syst Educ 32(1):1–8

Waszkowski R (2019) Low-code platform for automating business processes in manufacturing. IFAC-PapersOnLine 52(10): 376–381. https://doi.org/10.1016/j.ifacol.2019.10.060

Weyuker EJ (1988) Evaluating software complexity measures. EEE Trans Softw Eng 14(9):1357–1365. https://doi.org/10.1109/32.6178

Woo M (2020) The rise of no/low code software development—no experience needed? Engineering 6(9):960–961. https://doi.org/10.1016/j.eng.2020.07.007

Wulf F, Westner M, Strahringer S (2022) We have a platform, but nobody builds on it—what influences platform-as-a-service post-adoption? Int J Inform Syst Proj Manag 10(1):49–70. https://doi.org/10.12821/ijispm100103

Xiang Y, Pan W, Jiang H, Zhu Y, Li H (2019) Measuring software modularity based on software networks. Entropy 21(4):344. https://doi.org/10.3390/e21040344

Zuse H (2019) Software complexity: measures and methods. Vol. 4, Walter de Gruyter