HMD Praxis der Wirtschaftsinformatik (2024) 61:1295–1316
https://doi.org/10.1365/s40702-024-01096-x

SCHWERPUNKT

ADOxx: Eine Low-Code-Plattform für die Entwicklung von Modellierungswerkzeugen

Alexander Völz · Danial M. Amlashi · Patrik Burzynski · Wilfrid Utz

Eingegangen: 15. Februar 2024 / Angenommen: 7. Juli 2024 / Online publiziert: 2. August 2024

Zusammenfassung In dem semantisch reichhaltigen Gebiet der konzeptionellen Modellierung besteht ein Bedarf an offenen und benutzerfreundlichen Plattformen zur Entwicklung von Modellierungswerkzeugen. Dieser Bedarf ergibt sich aus dem stetigen Wandel von etablierten Modellierungsstandards (z. B. BPMN, UML, ArchiMate) sowie domänenspezifischen Anforderungen. In der Anwendung manifestieren sich derartige Anpassungen bei der Realisierung von Unternehmensinformationssystemen, die sich auch angesichts technischer Herausforderungen kontinuierlich weiterentwickeln. Dieser Beitrag präsentiert einen konzeptionellen Low-Code-Ansatz für Open-Source-Metamodellierungsplattformen, der auf Basis von ADOxx prototypisch umgesetzt ist. Hierbei spielen Low-Code-Konzepte wie visuelle Drag-and-Drop-Editoren, Point-and-Click-Interaktionen, sowie das Prinzip der Wiederverwendung eine zentrale Rolle. Letzteres ermöglicht eine effiziente Integration von unterschiedlichsten Komponenten existierender Implementierungen, die auf Basis von über 50 bereits entwickelten Modellierungswerkzeugen zusammengetragen wurden. Die entsprechenden Anforderungen, die sich für den vorgestellten Ansatz ergeben, basieren auf einem evolutionären Entwicklungsprozess, der sich aus den mit verschiedenen Versionen der Plattform entwickelten Werkzeugen ableitet.

✉ Alexander Völz · Danial M. Amlashi
Fakultät für Informatik, Universität Wien, Währinger Straße 29, 1090 Wien, Österreich
E-Mail: alexander.voelz@univie.ac.at

Danial M. Amlashi
E-Mail: danial.mohammadi.amlashi@univie.ac.at

Patrik Burzynski
OMiLAB NPO, Faulmanngasse 4, 1040 Wien, Österreich
E-Mail: patrik.burzynski@omilab.org

Wilfrid Utz
OMiLAB NPO, Lützowufer 1, 10785 Berlin, Deutschland
E-Mail: wilfrid.utz@omilab.org

Der Anspruch an Low-Code-Metamodellierungsplattformen ist die intuitive Realisierung von domänenspezifischen Modellierungswerkzeugen und die gleichzeitige Etablierung neuer Entwicklungsverfahren in der konzeptionellen Modellierung. Die Verwirklichung dieses erhobenen Anspruches steht im Einklang mit den aktuellen Trends im Bereich von Low-Code-Entwicklungsplattformen, welche durch die fortlaufende Anpassung an fortgeschrittene Funktionalitäten – wie die Integration von externen Diensten und Sicherheitsmechanismen, die zuvor primär durch traditionelle Programmierung umgesetzt wurden – vorangetrieben werden.

Schlüsselwörter Low-Code-Entwicklungsplattform · LCDP · Modellierungsmethoden · DSMM · Modellierungswerkzeuge · Domänenspezifik · ADOxx · Metamodellierung

ADOxx: A Low-Code Platform for the Development of Modeling Tools

Abstract In the semantically rich field of conceptual modeling, there is a need for open and user-friendly platforms for the development of modeling tools. This need arises from the constant change in both established modeling standards (e.g., BPMN, UML, ArchiMate) and domain-specific requirements. In application, such adaptations manifest themselves in the realization of enterprise information systems, which are constantly evolving in the face of technical challenges. This article presents a conceptual low-code approach for open-source metamodeling platforms, which is prototypically implemented on the basis of ADOxx. Low-code concepts such as visual drag-and-drop editors, point-and-click interactions, and the principle of reuse play a central role here. The latter enables the efficient integration of a wide variety of components from existing implementations, which have been compiled on the basis of over 50 modeling tools that have already been developed. The corresponding requirements that arise for the approach presented are based on an evolutionary development process that is derived from the tools developed with different versions of the platform. The requirement for low-code metamodeling platforms is the intuitive realization of domain-specific modeling tools and the simultaneous establishment of new development processes in conceptual modeling. The realization of this claim is in line with current trends in the field of low-code development platforms, which are driven by the ongoing adaptation to advanced functionalities—such as the integration of external services and security mechanisms that were previously implemented primarily through traditional programming.

Keywords Low-Code Development Platform (LCDP) · Domain-Specific Modeling Methods (DSMM) · Modeling Tools · ADOxx · Metamodeling

1 Einleitung

Informationstechnologien haben sich als entscheidender Treiber in zahleichen Bereichen etabliert, die vom stetigen und immer häufigeren Wandel dieser Technologien

beeinflusst sind. In jüngster Vergangenheit zeichnen sich so neue Herausforderungen hinsichtlich der Gewährleistung einer kontinuierlichen Interoperabilität ab, welche besonders im Kontext von Internet of Things (IoT), Cyber-Physical Systems (CPS), und Large Language Models (LLM) an Relevanz gewinnt. Darüber hinaus sind auch etablierte Modellierungsstandards wie BPMN und UML von dem Technologiewandel betroffen, wie deren iterative Evolution verdeutlicht (Platt und Thompson 2015; Geiger et al. 2016).

Aus dieser dynamischen und technologie-geprägten Vernetzung von unterschiedlichen Bereichen resultiert ein Bedarf an benutzerfreundlichen Entwicklungsplattformen, die Personen ohne nötige Kenntnisse eine Realisierung von Endbenutzeranwendungen ermöglichen. Dahingehend haben Low-Code/No-Code (LCNC) Ansätze, welche Fachexperten:innen ohne Programmierkenntnisse eine intuitive Entwicklungsumgebung zur Realisierung von Anwendungsideen bieten (Sahay et al. 2020), an Bedeutung gewonnen. Die Prinzipien der resultierenden Ansätze lassen sich auch auf den Entwicklungsprozess von Modellierungsmethoden ausweiten, der die Verwendung einer Metamodellierungsplattform durch geschulte Methodenentwickler:innen voraussetzt.

Der vorliegende Beitrag zielt darauf ab, Low-Code-Konzepte im Rahmen der Metamodellierungsplattform ADOxx[1] anzuwenden, um zu demonstrieren, wie der vorgestellte Ansatz eine schnelle Implementierung von neuen sowie eine iterative Anpassung von bestehenden Modellierungsmethoden ermöglicht und somit zugänglicher macht. Der evolutionäre Entwicklungsprozess, der dieser spezifischen Plattform zugrunde liegt, unterstützt sowohl die agile Reaktion auf technologische Herausforderungen als auch die Etablierung innovativer Entwicklungsverfahren. Durch die Integration von Low-Code-Methoden in die Metamodellierung wird so ein Entwicklungsparadigma geschaffen, das es ermöglicht, zeitnah auf sich ändernde Anforderungen zu reagieren und gleichzeitig robuste Modellierungswerkzeuge effizient bereitzustellen.

Zu diesem Zweck wird in Kap. 2 zunächst der theoretische Hintergrund beleuchtet, der sich mit LCNC Plattformen, Metamodellierung und Modellierungsmethoden sowie verwandten Werken auseinandersetzt. Darauffolgend wird der konzeptionelle Low-Code-Ansatz für Metamodellierungsplattformen präsentiert, dessen prototypische Umsetzung anhand von ADOxx veranschaulicht wird. Anschließend werden die abgeleiteten Erkenntnisse in Kap. 4 diskutiert, bevor abschließend die zusammengefassten Ergebnisse dieses Beitrags dargelegt werden.

2 Theoretischer Hintergrund

Im Folgenden werden theoretische Grundlagen erläutert, die für das Verständnis des vorgestellten Forschungsziels relevant sind. Dies umfasst LCNC Plattformen und den Bereich der Metamodellierung, die Basis der Entwicklung von Modellierungsmethoden.

[1] Verfügbar unter https://www.adoxx.org/.

2.1 LCNC-basierte Plattformen

Der Bedarf für LCNC-basierte Anwendungsentwicklung lässt sich zumindest teilweise auf die Verwendung von sogenannten *shadow tools* in Unternehmen zurückführen. Dabei handelt es sich um inoffizielle Anwendungen, die für individuelle Bedürfnisse genutzt werden und eine entscheidende Rolle in alltäglichen Geschäftsabläufen einnehmen (Handel und Poltrock 2011). Mit dem Ziel, dem damit einhergehenden Kontrollverlust entgegenzuwirken, haben Unternehmen begonnen Plattformen anzubieten, die eine interne Entwicklung von Anwendungen durch Personen ohne technisches Fachwissen, sogenannte *Citizen Developer,* ermöglichen (Sahay et al. 2020). Häufig lassen sich diese Werkzeuge auf altbewährte Prinzipen der modellgesteuerten Entwicklung zurückführen (Bock und Frank 2021; Lethbridge 2021), die einen visuellen Ansatz für die Realisierung von Anwendungen durch das Abstrahieren herkömmlicher Codierung bieten. Seither werden das Konzept des *Citizen Development* und die damit verbundenen LCNC-Plattformen auf Anwendungen ausgeweitet, die komplexe Datenanalysen und Automatisierungsprozesse umfassen. Die Einführung von LCNC-Plattformen verspricht eine innovative Bereitstellung von Unternehmensanwendungen, wodurch Strategien für eine effektive Einbindung dieser Plattformen notwendig werden (Käss et al. 2023).

Existierende Plattformen sind jedoch häufig nicht in der Lage, etablierte Praktiken spezifischer Bereiche ausreichend zu unterstützen, wodurch generierte Anwendungen auf äußerst umfangreichem und komplexem Code basieren (Lethbridge 2021). Zudem wird in einer Gegenüberstellung von Low-Code- und No-Code-Prinzipien hervorgehoben, dass diese sich primär durch den Grad der Reduzierung händischer Programmierung sowie die damit einhergehende Einschränkung der Personalisierungsmöglichkeiten voneinander abgrenzen (da Cruz et al. 2021). Vor diesem Hintergrund fokussieren wir uns im Rahmen der vorliegenden Arbeit auf das Konzept der *Low-Code Development Platform* (LCDP), da unser formuliertes Ziel eines konzeptionellen Low-Code-Ansatzes für Metamodellierungsplattformen einen gewissen Grad an Personalisierung voraussetzt. Dies führt im Umkehrschluss dazu, dass grundlegende Kenntnisse hinsichtlich der verwendeten Sprache zur Realisierung von Modellierungsmethoden (vgl. ADOSCRIPT in Abschn. 2.2) weiterhin notwendig sind. Die nachfolgenden Unterkapitel setzen sich näher mit generischen Architekturen und Eigenschaften von LCDPs auseinander.

2.1.1 Architektur von LCDPs

Eine generische Architektur von LCDPs wird in der Literatur nur vereinzelt aufgegriffen, was durch den meist anwendungsspezifischen Fokus solcher Plattformen erklärt werden kann. Die regelmäßig genannten Anwendungsfelder reichen dabei von mobilen Apps über Websites bis hin zu Systemen zur Verwaltung von Datenbanken und Geschäftsprozessen (Sahay et al. 2020; Bock und Frank 2021; Pinho et al. 2023; Rokis und Kirikova 2023). In der Arbeit von da Cruz et al. (2021) wird hingegen eine generische LCDP Architektur vorgestellt, die sich an dem sequentiellen Entwicklungsablauf von Low-Code-Anwendungen orientiert. Sahay et al. (2020) haben eine andere Perspektive gewählt und eine generische LCDP Architektur auf vier Ebenen

dargestellt: die interaktionsbasierte Anwendungsebene, auf der *Citizen Developer* Verhalten und Schnittstellen der jeweiligen Anwendung definieren; die Serviceintegrationsebene für die Anbindung externer APIs; die Datenintegrationsebene für die Verwaltung verschiedener Datenquellen; und die Implementierungsebene für die letztendliche Bereitstellung der Anwendung.

2.1.2 Eigenschaften von LCDPs

Im Vergleich zu generischen Architekturen von LCDPs haben deren funktionale Eigenschaften größere Aufmerksamkeit erhalten, insbesondere im Zuge von detaillierten Vergleichen existierender Plattformen (Sahay et al. 2020; Bock und Frank 2021) sowie umfassenden Literaturanalysen (Pinho et al. 2023; Rokis und Kirikova 2023). Vor dem Hintergrund der Existenz dieser ausführlichen Ausarbeitungen konzentrieren wir uns auf einen repräsentativen Ausschnitt derjenigen Eigenschaften, die mehrheitlich in den genannten Werken dokumentiert sind und gleichzeitig einen anwendungsagnostischen Fokus aufweisen. Die resultierende Übersicht unterteilt sich in breitgefasste Kategorien, denen zusammengetragene Eigenschaften zugeordnet sind (vgl. Tab. 1).

In den fünf berücksichtigten Arbeiten wurde einheitlich das grafische bzw. Drag-and-Drop-basierte Anwendungsdesign sowie die damit einhergehende Wiederverwendung vorgefertigter Komponenten thematisiert. Interoperabilität und Integration von externen APIs und Datenquellen ist besonders im Kontext aktueller technologischer Fortschritte von großer Bedeutung. Darüber hinaus wird die Anpassungsmöglichkeit existierender Komponenten der jeweiligen Plattform als wichtige Eigenschaft hervorgehoben. Die letzte der in Tab. 1 gelisteten Kategorien benennt die abschließende Bereitstellung der entwickelten Low-Code-Anwendung in der Cloud oder auf lokaler Infrastruktur.

Tab. 1 Zusammentragung meistgenannter Eigenschaften von LCDPs

Kategorie	Eigenschaft	Erwähnung in
Graphical User Interface (GUI) Design	Grafisches/Drag-and-Drop-basiertes Anwendungsdesign Vorgefertigte/Wiederverwendbare Komponenten (Formulare, Berichte Dashboards, Referenzmodelle)	Sahay et al. (2020) Bock und Frank (2021) da Cruz et al. (2021) Pinho et al. (2023) Rokis und Kirikova (2023)
Interoperabilität & Integration	Interoperabilität mit Schnittstellen von externen Anwendungen (APIs) Integration externer Datenquellen	Sahay et al. (2020) Bock & Frank (2021) Rokis und Kirikova (2023)
Anpassung & Erweiterung	Anpassung existierender Komponenten (Referenzmodelle, Datenmodelle, Prozesse) Erweiterung durch neue Komponenten für das GUI Design	Bock und Frank (2021) da Cruz et al. (2021) Rokis und Kirikova (2023)
Bereitstellung (Deployment)	Bereitstellung in der Cloud Bereitstellung auf lokaler Infrastruktur	Sahay et al. (2020) Pinho et al. (2023) Rokis und Kirikova (2023)

2.2 Metamodellierung und Modellierungsmethoden

Metamodellierung und die Entwicklung von Modellierungsmethoden sind grundlegende Aspekte der konzeptionellen Modellierung, die auf eine Formalisierung sowohl physischer als auch sozialer Sachverhalte abzielt, um Verständnis und Kommunikation zu fördern (Mylopoulos 1992). Im Zentrum dieser Disziplin steht das Prinzip der Abstraktion, das generell das Ausblenden von Einzelheiten zum Zweck einer vereinfachten Darstellung beschreibt (Mayr und Thalheim 2021). Ein gängiges Vorgehen ist die Verwendung von Modellierungssprachen, mit deren Hilfe komplexe Systeme durch diagrammatische Modelle repräsentiert werden können. Die Spezifikation solcher Sprachen beruht auf Metamodellen, die Modelle der Modellierungssprachen selbst sind und mit Hilfe von Metamodellierungssprachen konstruiert werden (Kühn et al. 1999; Karagiannis und Kühn 2002). Eine Konzeptualisierung der relevanten Elemente derartiger Sprachen findet sich in der Arbeit von Karagiannis und Kühn (2002), in der das *Generic Modeling Method Framework* (GMMF) vorgestellt wird. Gemäß dem GMMF muss eine Modellierungssprache eine einheitlich definierte *Syntax, Notation* und *Semantik* bieten, um sicherzustellen, dass die entwickelten Modelle sowohl von Menschen interpretiert als auch von Maschinen verarbeitet werden können. Darüber hinaus spielen *Mechanismen & Algorithmen*, die Fähigkeiten einer Sprache um Funktionen wie Modelltransformationen und Integration mit anderen Systemen erweitern, eine entscheidende Rolle. Zusammen mit dem *Modellierungsverfahren* bilden diese drei Elemente eine Modellierungsmethode.

In diesem Zusammenhang haben domänenspezifische Modellierungsmethoden (DSMMs) an Relevanz gewonnen, die eine Anpassung von domänenspezifischen Sprachen (DSLs) sowie *Mechanismen & Algorithmen* an den jeweiligen Untersuchungskontext ermöglichen (Karagiannis et al. 2016, 2022). Eine anschließende Bereitstellung solcher Methoden in Form eines Modellierungswerkzeuges vereinfacht die Nutzung und Weitergabe, indem ein gebrauchsfertiges Installationspaket angeboten wird. Die Anwendungsbereiche der resultierenden Werkzeuge inkludieren unter anderem Design Thinking, Japanische Kreativdienste, plattformübergreifende IoT Anwendungen, Lehrmittel und viele weitere.[2] Für die Realisierung von DSLs und DSMMs ist sowohl eine anwendungsorientierte Metamodellierungssprache als auch ein richtungsweisendes Vorgehensmodell notwendig. Hinsichtlich Letzterem sind bereits verschiedene Ansätze etabliert worden, die einen strukturierten und iterativen Ablauf für das Erstellen von DSLs und DSMMs definieren (Frank 2013; Karagiannis 2015; Michael und Mayr 2015).

Vor dem Hintergrund einer anwendungsorientierten Metamodellierungssprache ist jüngst ein neues Paradigma von Karagiannis (2024) vorgestellt worden, das Entwicklern bei der Methodenrealisierung unterstützen soll: eine *domänenspezifische Sprache für die Realisierung von Modellierungsmethoden* (MM-DSL). Am Beispiel von ADOSCRIPT, einer MM-DSL Instanz, erweitert dieses Paradigma das

[2] Die genannten Anwendungsbereiche basieren auf Modellierungsmethoden, die mit der Open-Source-Metamodellierungsplattform ADOxx realisiert wurden und frei verfügbar sind (weiterführende Informationen finden sich in Karagiannis et al. (2016; 2022), oder unter: https://www.omilab.org/activities/projects/).

konventionelle Konzept domänenspezifischer Sprachen, um den Anforderungen der konzeptionellen Modellierung gerecht zu werden. Speziell geht es dabei um Anforderungen an die Spezifikation von Modellierungsmethoden innerhalb des GMMF (Karagiannis und Kühn 2002). In diesem Kontext stellt ADOSCRIPT eine implementierungsorientierte MM-DSL Instanz dar, durch deren Nutzung bereits eine Vielzahl von Modellierungsmethoden entwickelt wurden (Karagiannis et al. 2016, 2022). In dem Prozess der Realisierung dieser Methoden sind vier fundamentale Metakonzepte wegweisend, die durch ADOSCRIPT operationalisiert werden und in Tab. 2 zusammengefasst sind.

Instanziierung beschreibt die Metamodell-Spezifikation einer Modellierungsmethode, indem das entsprechende Meta²Modell der verwendeten MM-DSL instanziiert wird.

Visualisierung befasst sich mit der grafischen Darstellung der Konzepte des erstellten Metamodells und wie diese auf einer visuellen Ebene miteinander interagieren, sodass deren benutzerfreundliche und verständliche Darstellung ermöglicht wird.

Individualisierung konzentriert sich auf die Anpassung vorgefertigter Strukturen und Funktionen an die besonderen Anforderungen der verschiedenen Anwendungsbereiche hinsichtlich Datenstrukturen, Interaktion und Integration.

Umsetzung stellt individuelle Erweiterungsmöglichkeiten durch Scripting-Funktionen zur Verfügung, sodass Modelltransformationen, Benutzerinteraktionen und Integration mit externen Systemen anwendungsspezifisch entwickelt werden können.

Hilfskonzepte sind nicht in Tab. 2 inkludiert, weil diese für eine Operationalisierung auf Metamodellierungsplattformen nicht zwangsläufig notwendig sind. Dennoch können externe Hilfskonzepte die Interoperabilität und Bereitstellung von Modellierungsmethoden fördern.

2.3 Verwandte Arbeiten

Obwohl sich ein Großteil der existierenden Anwendungsfelder von LCDPs auf den Software Engineering Kontext konzentriert (vgl. Abschn. 2.1.1), gilt es dennoch ausgewählte Werke hervorzuheben, die eine Verbindung zwischen Low-Code-Ansätzen und Methodenentwicklung herstellen.

Eine der wenigen Auseinandersetzungen mit der Nutzung von Low-Code-Paradigmen für die Methodenentwicklung findet sich in de Olivera et al. (2022). Die

Tab. 2 DSL Metakonzepte zur Realisierung von Modellierungsmethoden (Karagiannis 2024)

Metakonzept	Ermöglicht	Ergebnis
Instanziierung	Metamodell-Design	Metamodell
Visualisierung	Metamodell-Darstellung	Notation, Benutzeroberfläche, Datencontainer
Individualisierung	Metamodell-Erweiterung, Anpassung von Funktionalitäten	Domänenspezifische Semantik und Analysen
Umsetzung	Metamodell-gestützte Programmierung	Domänenspezifische und generische Operationen

Autoren argumentieren für die Vorteile dieser Paradigmen hinsichtlich der Vereinfachung des Erstellens und der Anpassung von Methoden. Der resultierende Ansatz und die darauf aufbauende *LOMET* (LOw-code Method Engineering Tool) Implementierung inkludiert einen visuellen Methoden-Editor, der sich auf die Definition von Prozessmodellen und entsprechenden Metamodellen beschränkt, was nach der Definition von Frank (2011) den minimalen Anforderungen einer Modellierungsmethode entspricht. Daher stellt diese Arbeit einen ersten Nachweis der Anwendbarkeit von LCDPs in der konzeptionellen Modellierung dar.

Für den Kontext des vorliegenden Beitrags ist außerdem die Arbeit von Ibrahimi und Moudilos (2022) relevant, in der sich die Autoren mit der Wiederverwendung von Modellen im Kontext von Low-Code-Plattformen auseinandersetzen. Dabei wird ein Ansatz auf Basis der *zAppDev* LCDP demonstriert, die das Erstellen einer großen Auswahl unterschiedlicher Anwendungen ermöglicht. In diesem Ansatz wird ein Graphen-basiertes Repository für das Empfehlen und Wiederverwenden von Modellierungselementen verwendet, ein Aspekt, der auch für den Bereich der Methodenentwicklung relevant ist.

Abschließend wird kurz auf die Arbeit von Rybiński und Smialek (2022) eingegangen, in der die Relevanz von domänenspezifischem Wissen und Regeln für LCDP thematisiert wird. Bereits im Zuge des umfassenden Vergleiches sieben existierender Plattformen von Bock und Frank (2021) ist angemerkt worden, dass domänenspezifische Funktionalitäten einen häufigen Mangel darstellen. Das Ziel der Arbeit ist eine automatisierte Generierung von Domänenlogik-Code, die durch Wiederverwendung von Domänenwissen ermöglicht wird (Rybiński und Smialek 2022). Der Ansatz der Wiederverwendung von domänenspezifischem Wissen wird auch im folgenden Verlauf dieser Arbeit angewendet.

3 Low-Code-Ansatz für Metamodellierungsplattformen

Das Konzept einer LCDP stellt einen transformativen Ansatz dar, der darauf abzielt, den Entwicklungsprozess von Endbenutzeranwendungen zu demokratisieren, indem er die Komplexität und den Umfang der erforderlichen händischen Programmierung reduziert. Darauf aufbauend ist das grundlegende Ziel der vorliegenden Arbeit, die gleichen Prinzipien auf den Entwicklungsprozess von Modellierungsmethoden anzuwenden.

3.1 Ansätze für die Entwicklung von Modellierungsmethoden

Für die Entwicklung von Modellierungsmethoden ist eine anwendungsorientierte MM-DSL notwendig, die eine Realisierung der vorgestellten Metakonzepte ermöglicht. Im Kontext der Arbeit von Karagiannis (2024) wird ADOSCRIPT als eine solche MM-DSL vorgestellt (vgl. Abschn. 2.2), die durch ihre generischen Konstrukte als plattformunabhängig gilt, auch wenn auf ihr basierende Entwicklungen bisher primär über die Metamodellierungsplattform ADOxx realisiert wurden. Unter der Berücksichtigung dieser essentiellen Elemente sind in Abb. 1 ein *Coding-*

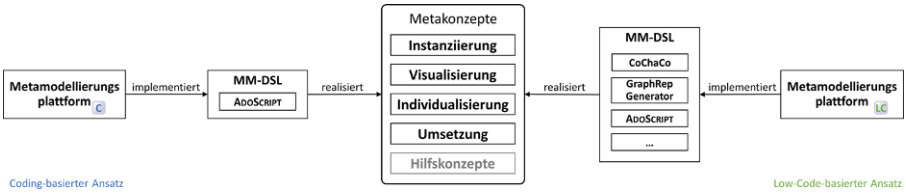

Abb. 1 Coding-basierter vs. Low-Code-basierter Ansatz für die Entwicklung von Modellierungsmethoden

basierter Ansatz sowie ein *Low-Code-basierter Ansatz* für die Entwicklung von Modellierungsmethoden dargestellt.

Coding-basierter Ansatz Dieser Ansatz entspricht dem traditionellen Vorgehen bei der Entwicklung von Modellierungsmethoden, das von der linken Seite in Abb. 1 ausgeht. Dabei wird die Spezifikation der jeweiligen Modellierungssprache sowie zugehöriger Mechanismen und Algorithmen mithilfe einer entsprechenden MM-DSL, in diesem Fall ADOSCRIPT, durch rein code-basierte Interaktionen umgesetzt. Das bedeutet, dass die Realisierung der grundlegenden Metakonzepte fortgeschrittene Kenntnisse der Methodenentwicklung hinsichtlich der verwendeten MM-DSL voraussetzt. Abschließend ist für diesen Ansatz eine Metamodellierungsplattform notwendig, die eine resultierende Implementierung der MM-DSL Spezifikation als Tool ermöglicht, indem die jeweilige Plattform den MM-DSL Code gemäß dem Prinzip eines Compilers in ein gebrauchsfertiges Modellierungswerkzeug übersetzt.[3]

Low-Code-basierter Ansatz Die iterative Evolution von Metamodellierungsplattformen, wie beispielsweise ADOxx, hat dazu geführt, dass individuelle Elemente im Prozess der Entwicklung von Modellierungsmethoden ohne weitreichende Erfahrung hinsichtlich der verwendeten MM-DSL realisiert werden können. Nichtsdestotrotz gilt es zu berücksichtigen, dass grundlegende Kenntnisse auf technischer und konzeptioneller Ebene weiterhin notwendig sind, damit neu generierte sowie widerverwendete Code-Komponenten sowohl überprüft als auch nach Bedarf angepasst werden können. Aus diesem Grund stellt ADOSCRIPT in dem in Abb. 1 dargestellten *Low-Code-basierten Ansatz* weiterhin einen relevanten Bestandteil der MM-DSL dar. Im Gegensatz zum *Coding-basierten Ansatz* muss die MM-DSL, genau wie die sie implementierende Metamodellierungsplattform, jedoch um gewisse Low-Code-Komponenten erweitert werden, die im nachfolgenden Unterkapitel näher erläutert werden.[4] Dies ermöglicht eine plattform-basierte Interaktion für die Entwicklung von Modellierungsmethoden, die gemäß ausgewählter Low-Code-Konzepte bereitgestellt und durch gewisse externe Hilfswerkzeuge gefördert wird. Das *Agile Modeling Method Engineering* (AMME) Vorgehensmodell steuert diese Entwicklung

[3] Basierend auf der limitierten Funktion der Metamodellierungsplattform im Kontext des *Coding-basierten Ansatzes* wird diese mit einem tiefgestellten *C* gekennzeichnet, das für *Coding* steht.

[4] Unter Berücksichtigung dieser Erweiterungen ist die resultierende Version der Metamodellierungsplattform mit einem *LC* als Abkürzung für *Low-Code* gekennzeichnet.

durch fünf richtungsweisende Phasen (*Create, Design, Formalize, Develop, Deploy*), die eine iterative Anpassung und Weiterentwicklung von DSMMs gemäß sich stetig wandelnden Anforderungen unterstützen (Karagiannis 2015). Im Zentrum des *Low-Code-basierten Ansatzes* stehen die grundlegenden Metakonzepte, die, genau wie im Kontext des *Coding-basierten Ansatzes,* durch die Nutzung einer anwendungsorientierten MM-DSL realisiert werden.

3.2 ADOxx als Low-Code-Entwicklungsplattform für Modellierungsmethoden

Die Open-Source-Metamodellierungsplattform ADOxx ist im Verlauf dieser Arbeit bereits als die primäre Umgebung für die Entwicklung von auf ADOSCRIPT basierenden Modellierungsmethoden vorgestellt worden. Die Vielfalt der unterschiedlichen Methoden und Werkzeugen, die so realisiert wurden, ist in Sammelbänden (Karagiannis et al. 2016, 2022), Metamodel-basierten Analysen (Bork 2018) und auch online durch ausführliche Projektbeschreibungen dokumentiert (vgl. Abschn. 2.2).

Im Folgenden wird die prototypische Umsetzung des vorgestellten Low-Code-Ansatzes für eine benutzerfreundliche Entwicklung von Modellierungsmethoden (vgl. Abschn. 3.1) anhand der ADOxx Plattform demonstriert. Dahingehend gilt es, die zuvor erwähnten Erweiterungen in Form von plattformintegrierten Funktionalitäten sowie externen Hilfswerkzeugen unter Berücksichtigung der jeweiligen Metakonzepte, deren Realisierung sie unterstützen, vorzustellen. In diesem Kontext ist außerdem die jeweilige Interaktion von Bedeutung, die eine Nutzung der Erweiterungen und Hilfswerkzeuge ermöglicht. Eine Übersicht der Zusammenhänge, die zwischen den bereits vorgestellten Metakonzepten, den darauf aufbauenden Interaktionen sowie zugehörigem Support, und den resultierenden Eigenschaften bestehen, ist in Tab. 3 dargestellt.

ADOxx-basierte Interaktion Die möglichen Interaktionen mit der ADOxx Plattform und externen Hilfswerkzeugen orientieren sich an etablierten Low-Code-Eigenschaften. Dabei spielen Point-and-Click-Interaktionen (Sahay et al. 2020), visuelle Drag-and-Drop-Editoren (Pinho et al. 2023) sowie das Prinzip der Wiederverwendung durch Copy-and-Paste-Ansätze (Rokis und Kirikova 2023) eine zentrale Rolle. Darüber hinaus werden auch schlichte Vorschau-Funktionen berücksichtigt. Diese vier Ansätze bilden in Tab. 3 die Grundlage der ADOxx-basierten Interaktionsmöglichkeiten.

ADOxx-basierter Support & unterstützte Eigenschaften Für die Realisierung der Metakonzepte stehen sowohl plattformintegrierte Funktionalitäten als auch externe Hilfswerkzeuge zur Verfügung, die unterschiedliche Eigenschaften für die Entwicklung von Modellierungswerkzeugen im Sinne eines Low-Code-Ansatzes bereitstellen. Diese Komponenten und die jeweiligen unterstützten Eigenschaften sind nachfolgend erläutert:

- Als *Modellierungsmethoden Design Editor* (MMDE) bezeichnen wir im Folgenden eine Zusammenfassung sämtlicher Funktionen und Eigenschaften, die zur Realisierung der Metakonzepte direkt über die Benutzeroberfläche der ADOxx

Tab. 3 ADOxx$_{(LC)}$ Interaktionen mit jeweiligem Support und dadurch unterstützte Eigenschaften gruppiert nach zugeordneten Metakonzepten

Metakonzept	ADOxx$_{(LC)}$-basierte Interaktion	ADOxx$_{(LC)}$-basierter Support	Unterstütze Eigenschaften
Instanziierung	Point-and-Click	MMDE	Point-and-Click-basiertes Metamodell-Design
	Drag-and-Drop	CoChaCo	Visuelles Metamodell-Design
Visualisierung	Vorschau	MMDE	Code-basierte Vorschau von Notation-Design
	Drag-and-Drop	GraphRep Generator	Drag-and-Drop-basiertes Notation-Design
	Copy-and-Paste	GraphRep Repository	Copy-and-Paste-basierte Notation-Wiederverwendung
Individualisierung	Point-and-Click	MMDE	Individualisierung vorgefertigter Analysemechanismen
			Individualisierung der Attributdarstellung
Umsetzung	Point-and-Click	MMDE	Interoperabilität mit externen Diensten und Datenquellen
			RDF-basierte Modell-/Metamodell-Transformation
	Vorschau Copy-and-Paste	MMDE	Vorschau und Ausführung existierender Funktionalitäten Copy-and-Paste-basierte Wiederverwendung von Funktionalitäten
Hilfskonzepte	Copy-and-Paste	AutoPDP	Automatisierte Bereitstellung als Modellierungswerkzeug
	Point-and-Click	OLIVE	Zugriff und Integration existierender Microservices Anpassung existierender Microservice-Konnektoren

Plattform angeboten werden. Für das Konzept der *Instanziierung* wird ein Point-and-Click-basierter Ansatz für das Erstellen von Metamodellen angeboten, bei dem Klassen, Attribute, sowie Beziehungen gemäß dem Meta²Modell der jeweiligen Metamodellierungsplattform (in unserem Beispiel ADOxx) intuitiv hinzugefügt und angepasst werden können. Ergänzend wird in diesem Kontext auch das externe Hilfswerkzeug *CoChaCo* (Karagiannis et al. 2019) angeboten, das ein visuelles Erstellen von Metamodellen gemäß dem Prinzip von Drag-and-Drop-Interaktionen ermöglicht. Hinsichtlich der *Visualisierung* stehen zwei externe Hilfswerkzeuge zur Verfügung (*GraphRep Generator, GraphRep Repository*), die nachfolgend erläutert werden. Vor diesem Hintergrund bietet ADOxx die Möglichkeit, selbst geschriebenen oder wiederverwendeten GraphRep[5] Code zur Überprüfung des Resultats visuell innerhalb der Entwicklungsumgebung darzustellen. Zur Realisierung des Metakonzepts *Individualisierung* bietet die ADOxx

[5] GraphRep (Graphical Representation) wird innerhalb der MM-DSL ADOSCRIPT zur ADOxx-spezifischen Realisierung der visuellen Repräsentation von Klassen und Beziehungen in einem Metamodell genutzt (siehe https://www.adoxx.org/documentation/graphrep/).

Plattform vorgefertigte Analysemechanismen, die gemäß dem Point-and-Click-Prinzip einen Einrichtungsassistenten (auch *Wizzard* genannt) zur Verfügung stellen, der eine domänenspezifische Anpassung dieser Mechanismen ermöglicht. Zusätzlich wird eine benutzerfreundliche Anpassung der Darstellung von Attributen auf sowohl konzeptioneller als auch visueller Ebene angeboten. Das Metakonzept *Umsetzung* ist von großer Bedeutung für das Erstellen von domänenspezifischen Funktionalitäten, die über die Fähigkeiten einer rein visuellen Modellierungssprache hinausgehen. Dahingehend bietet ADOxx eine Vielzahl an Möglichkeiten um Interoperabilität mit externen Diensten und Datenquellen sicherzustellen. Neben fortgeschrittenen Verfahren zur Etablierung solcher Funktionalitäten, die in der Regel das Schreiben von ADOSCRIPT Code voraussetzen, existieren auch Ansätze um bereits implementierte Microservices des *OLIVE* Frameworks (Falcioni und Woitsch 2021) ohne entsprechende Erfahrungen zunächst für den Zweck der jeweiligen Methode anzupassen und anschließend durch spezielle Konnektoren zu integrieren. Darüber hinaus gibt es generische Funktionalitäten, die für eine Verwendung innerhalb der ADOxx Plattform als Erweiterungen hinzugefügt werden können, sodass eine Transformation von Modellen und Metamodellen in ein RDF-basiertes Format ermöglicht wird (Buchmann und Karagiannis 2016; Karagiannis und Buchmann 2016). Eine weitere generische Funktionalität, die der Plattform hinzugefügt werden kann, ist das Anzeigen sowie prototypische Ausführen des ADOSCRIPT Codes von Mechanismen und Algorithmen, die im Zuge der über 50 mit ADOxx entwickelten Modellierungsmethoden bereits realisiert wurden. Gemäß dem Prinzip der Wiederverwendung kann so eine effiziente Integration existierender Implementierungen und Funktionalitäten gewährleistet werden.

- *CoChaCo*[6] ist ein Hilfswerkzeug, das ein Design von Metamodellen mithilfe visueller Drag-and-Drop-Interaktionen ermöglicht und somit eine externe Erweiterung der MMDE Funktionen darstellt (Karagiannis et al. 2019).
- Der *GraphRep Generator*[6] ist ein externer Service, der das Erstellen von Vektorgrafiken in einem visuellen Editor sowie deren anschließende Transformation in *GraphRep* Code ermöglicht.
- Das *GraphRep Repository*[6] umfasst eine Sammlung von grafischen Repräsentationen, die aus verschiedenen Projekten zusammengetragen wurden. Nach dem Copy-and-Paste-Prinzip können so bereits existierende Implementierungen wiederverwendet und nach Bedarf angepasst werden.
- *AutoPDP*[6] (Auto Product Development Packaging) ist ein externer Service, der Methodenentwicklern:innen ein einfaches Vorgehen bietet um ADOxx-basierte Modellierungsmethoden in anwendungsfertige Modellierungswerkzeuge umzuwandeln. Dies inkludiert, dass die ursprüngliche Methode in einem automatisierten Vorgehen durch produktrelevante Informationen erweitert und als Installationspaket gemäß dem Open-Source-Prinzip bereitgestellt wird.
- *OLIVE*[6] ist ein Framework, das sowohl für das Erstellen neuer als auch die Konfiguration bestehender Microservices verwendet wird (Falcioni und Woitsch 2021). Die daraus resultierenden Funktionalitäten weisen häufig einen direkten Bezug zu generischen sowie spezifischen Komponenten des GMMF auf.

[6] Verfügbar unter: https://www.adoxx.org/lowcode/.

3.3 Anwendung der ADOxx Low-Code-Entwicklungsplattform

Zur Verdeutlichung des vorgestellten Low-Code-Ansatzes wird nachfolgend ein Beispiel herangezogen, das bereits in früheren Arbeiten thematisiert wurde: der *Drone Tour Guide Case* (Muck et al. 2022; Voelz et al. 2023). Das Beispiel illustriert exemplarisch, wie die grundlegenden Metakonzepte durch Low-Code-basierte Interaktionen im Kontext der ADOxx Entwicklungsplattform realisiert werden können. Im Zuge der praktischen Vorführung dieser Low-Code-basierten Entwicklung von Modellierungsmethoden wird auch eine Gegenüberstellung zu traditionellen, Coding-basierten Ansätzen vorgenommen. Es gilt dabei anzumerken, dass wir uns im Rahmen des ausgewählten Beispiels auf die Perspektive des Methodenentwicklers konzentrieren, während der Zweck der Methode und des Erstellens von entsprechenden Modellinstanzen zweitrangig ist und daher nicht näher erläutert wird. Grundlegend ermöglicht die Methode das Erstellen von touristischen Touren als konzeptionelle Modelle. Im verbleibenden Verlauf des Kapitels wird anhand dieses Beispiels verdeutlicht, wie die Realisierung der grundlegenden Metakonzepte im Kontext der ADOxx Plattform durch den vorgestellten Ansatz unterstützt wird.

Instanziierung In den nachfolgenden Erläuterungen wird angenommen, dass die einleitenden *Create, Design*, und *Formalize* Phasen des AMME Vorgehensmodells (Karagiannis 2015) abgeschlossen sind, wodurch relevante Konzepte, Anforderungen, und spezifisches Domänenwissen bereits identifiziert und formalisiert sind. Im nächsten Schritt, der *Develop* Phase, geht es im Zuge der *Instanziierung* darum, das jeweilige Meta²Model der verwendeten MM-DSL gemäß einem spezifizierten Metamodell zu instanziieren. In Abb. 2 sind in Anlehnung an Abb. 1 der Coding-basierte und der Low-Code-basierte Ansatz hinsichtlich der Instanziierung des Meta²Models der ADOxx Plattform gegenübergestellt. Ersterer ist auf der linken Seite durch einen

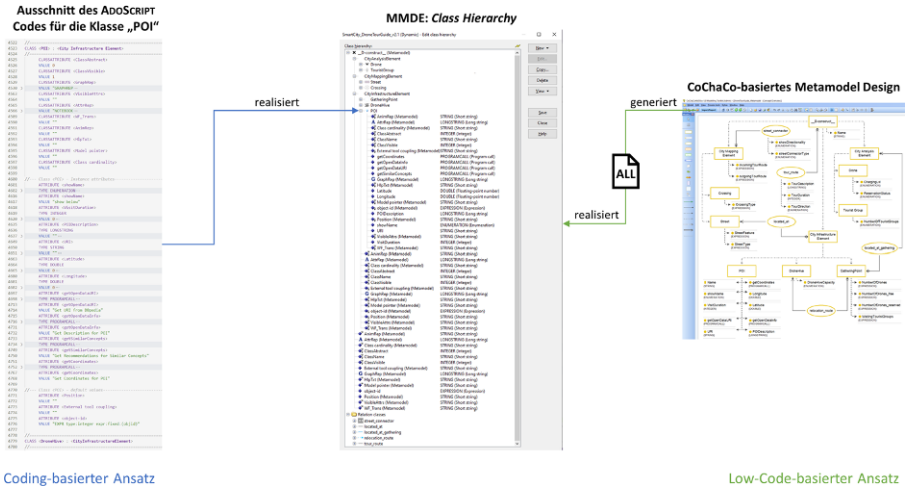

Abb. 2 Gegenüberstellung des Coding-basierten und Low-Code-basierten Ansatzes für das Design von Metamodellen (Instanziierung)

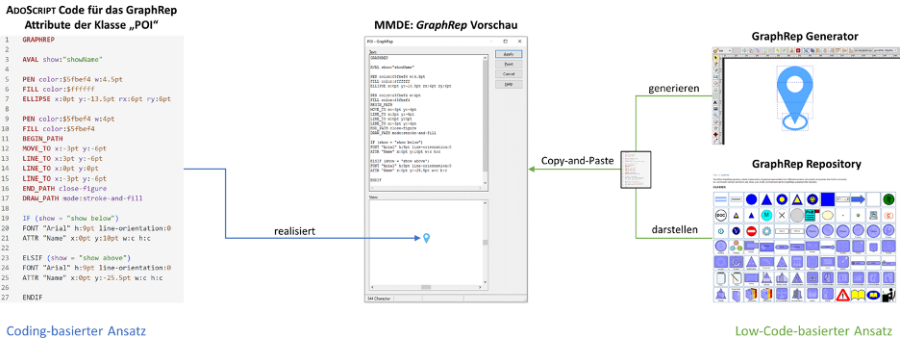

Abb. 3 Gegenüberstellung des Coding-basierten und Low-Code-basierten Ansatzes für das Realisieren der grafischen Darstellung von Metamodell-Konzepten (Visualisierung)

Ausschnitt des ADOSCRIPT Codes dargestellt, der die Klasse „POI" (Point of Interest) realisiert. Dem gegenüber steht die Verwendung des Hilfswerkszeugs *CoChaCo*, das ein Drag-and-Drop-basiertes Design von Metamodellen durch grundlegende Konzepte ermöglicht. Im Anschluss kann durch integrierte Funktionalitäten eine entsprechende Datei im ALL Format[7] generiert werden, die zur Realisierung des erstellten Metamodells unter Berücksichtigung von Plattform-spezifischen Anforderungen dient. Entsprechend ist in der Mitte von Abb. 2 das realisierte Metamodell im Kontext der *Class Hierarchy*[8] des MMDE visualisiert, der ein Design und Adaptieren von Metamodellen in der ADOxx Umgebung durch intuitive, Point-and-Click-basierte Interaktionen bietet.

Aus der Perspektive von Methodenentwicklern:innen, die das Erstellen von touristischen Touren auf der Basis von ADOxx ermöglichen wollen, stehen zusammenfassend drei Ansätze zur Verfügung: das händische Schreiben von Plattform-spezifischem ADOSCRIPT Code, das Drag-and-Drop-basierte Designen von Metamodellen mithilfe von *CoChaCo*, und der Point-and-Click-basierte Ansatz innerhalb der *Class Hierarchy* des MMDE.

Visualisierung Nach der syntaktischen Spezifikation der Modellierungssprache gilt es, die jeweiligen Konzepte unter Berücksichtigung der semantischen Bedeutung visuell darzustellen. Entsprechend formt die *Visualisierung* der Syntax gemäß dem GMMF (Karagiannis und Kühn 2002) einen essentiellen Bestandteil bei der Entwicklung von Modellierungsmethoden. Dahingehend sind in Abb. 3 eine Gegenüberstellung eines Coding-basierten und Low-Code-basierten Ansatzes zur Realisierung einer grafischen Darstellung veranschaulicht. Auf der linken Seite ist der *GraphRep* Code dargestellt, der ein blaues Standortsymbol als visuelle Repräsentation für die Klasse „POI" realisiert.[9] Alternativ stehen zwei Low-Code-basierte

[7] ALL (ADOxx Library Language) wird innerhalb der MM-DSL ADOSCRIPT zur Realisierung von Libraries genutzt (siehe https://www.adoxx.org/documentation/all/).

[8] Hierbei wurden repetitive Elemente des Metamodells zur besseren Übersicht teilweise ausgeblendet.

[9] Für eine ADOSCRIPT-basierte Realisierung innerhalb der ADOxx Plattform wird die *GraphRep* Spezifikation als String Wert des zugehörigen Klassenattributes definiert (vgl. Code-Zeile 4530 in Abb. 2).

Abb. 4 Gegenüberstellung des Coding-basierten und Low-Code-basierten Ansatzes für das Individualisieren von Analysemechanismen (Individualisierung)

Ansätze zur Verfügung. Zum einen können Visualisierungen durch die Anordnung von Vektorgrafiken innerhalb des *GraphRep Generator* entworfen werden, der eine automatisierte Generierung von *GraphRep* Code anbietet. Zum anderen können bestehende Visualisierungen aus dem *GraphRep Repository* wiederverwendet werden. In beiden Fällen muss der generierte oder wiederverwendete Code nach dem Copy-and-Paste-Prinzip innerhalb des MMDE eingefügt werden. Der MMDE bietet dahingehend eine Vorschau des jeweiligen *GraphRep* Codes an (siehe Abb. 3).

Gemäß diesen Ansätzen ergeben sich für eine:n Methodenentwickler:in unterschiedliche Vorgehensweisen für das Realisieren von grafischen Darstellungen der Konzepte eines entwickelten Metamodells, je nach den vorhandenen Kenntnissen bezüglich ADOSCRIPT. Der *GraphRep Generator* und das *GraphRep Repository* ermöglichen eine Generierung bzw. Wiederverwendung von etablierten Visualisierungen, sodass keinerlei Vorwissen notwendig ist. Innerhalb des MMDE können anschließend individuelle Anpassungen vorgenommen werden, vorausgesetzt grundlegende ADOSCRIPT Kenntnisse sind vorhanden. So können unter anderem Attributspezifische Visualisierungen realisiert werden, wie bspw. die Platzierung des POI Namens unter oder über dem Standortsymbol.[10]

Individualisierungen Die ADOxx Plattform bietet eine Vielzahl von Möglichkeiten an, um Attributdarstellungen zu individualisieren.[11] In diesem Abschnitt konzentrieren wir uns hingegen auf zwei verschiedene Ansätze hinsichtlich der *Individualisierung* von vorgefertigten Analysemechanismen, die in Abb. 4 gegenübergestellt sind. Der Coding-basierte Ansatz ist auf der linken Seite durch einen Ausschnitt einer Spezifikation dargestellt, die eine AQL[12] Abfrage zum Identifizieren von Objekten der Klasse „GatheringPoint" mit verfügbaren Dronen realisiert. Der wenig intuitive AQL Code der Abfrage kann hingegen auch mithilfe eines Einrichtungsassistenten gemäß dem Point-and-Click-Prinzip realisiert werden, wie im rechten Teil von Abb. 4 visualisiert ist.

[10] Vgl. Codezeilen 3 sowie 19-27 des Coding-basierten Ansatzes in Abb. 3.

[11] Derartige Individualisierungen werden durch das ADOxx-spezifische Attribut *AttrRep* (Attribute Representation) realisiert (siehe https://www.adoxx.org/documentation/attrrep/).

[12] AQL (ADOxx Query Language) wird innerhalb der MM-DSL ADOSCRIPT zur Realisierung von Modell- und Metamodell-gestützten Analysemechanismen genutzt (siehe https://www.adoxx.org/documentation/aql/).

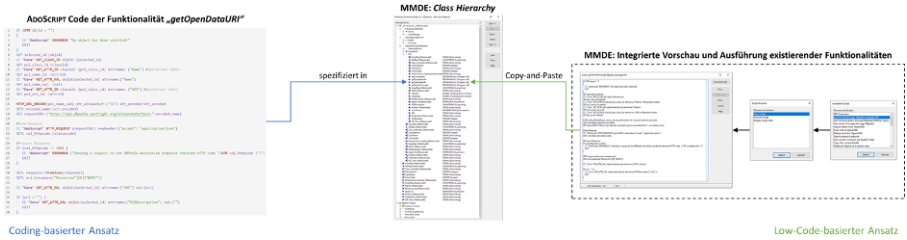

Abb. 5 Gegenüberstellung des Coding-basierten und Low-Code-basierten Ansatzes zur Realisierung von Metamodell-gestützten Funktionalitäten (Umsetzung)

Hinsichtlich der *Individualisierung* von Analysemechanismen stellt besonders das Low-Code-basierte Editieren von Abfragen aufgrund der speziellen AQL Syntax einen Mehrwert für Methodenentwickler:innen dar, die wenig Vorkenntnisse bezüglich AdoScript besitzen. So können komplexe Analysen konstruiert und für Nutzer:innen der Modellierungsmethode in einer benutzerfreundlichen Weise verfügbar gemacht werden.

Umsetzung Die *Umsetzung* von Metamodell-gestützten Funktionalitäten im Sinne eines Low-Code-Ansatzes anzubieten stellt eine Herausforderung dar, weil in der Regel fortgeschrittene AdoScript Kenntnisse in diesem Kontext notwendig sind. Die einfache Interoperabilität mit externen Diensten (bspw. eine konfigurierbare HTTP-Abfrage ausführt) und der RDF-basierten Transformation von Modellen sowie Metamodellen (vgl. Tab. 3) können jedoch auch ohne Vorkenntnisse genutzt werden. In diesem Abschnitt konzentrieren wir uns auf eine prototypische Erweiterung, die eine Wiederverwendung existierender Funktionalitäten unter der Annahme grundlegender AdoScript Kenntnisse ermöglicht. Vor diesem Hintergrund ist auf der linken Seite von Abb. 5 die Coding-basierte Realisierung der Funktionalität „getOpenDataURI" dargestellt, die eine HTTP Anfrage an die DBpedia Spotlight API (Mendes et al. 2011) mit dem Wert des Attributes „Name" der jeweiligen POI Instanz als Input ausführt und die Antwort als Wert des Attributes „URI" speichert. Der Low-Code-basierte Ansatz in Form einer prototypischen Erweiterung baut darauf auf, dass solche etablierten Funktionalitäten in einem Repository gesammelt werden, um diese während der Entwicklung zukünftiger Modellierungsmethoden verfügbar zu machen. Unter der Voraussetzung grundlegender AdoScript Kenntnisse kann die „getOpenDataURI" Funktionalität durch wenige Anpassungen wiederverwendet werden, um DBpedia URIs für beliebige Klassen abzurufen. Darüber hinaus bietet das Repository an existierenden Funktionalitäten Methodenentwicklern:innen einen Einblick in bestehende Umsetzungen, die als Inspiration für das Etablieren in neuen Anwendungsfeldern dienen kann.

Zusammenfassend kann festgestellt werden, dass die ADOxx Plattform eine Reihe an Low-Code-basierten Interaktionen anbietet, die eine Realisierung der grundlegenden Metakonzepte auch ohne weitreichende AdoScript Kenntnisse ermöglicht. Anhand des Anwendungsbeispiels konnte verdeutlicht werden, wie die resultierenden Ansätze im Kontext eines konkreten Methodenentwicklungsprozesses zur Anwendung kommen. Im Rahmen der abschließenden *Deployment* Phase des

AMME Vorgehensmodells steht außerdem der externe *AutoPDP* Service zur Verfügung, um entwickelte Methoden benutzerfreundlich als anwendungsfertige Modellierungswerkzeuge bereitstellen zu können, bevor eine weitere AMME Iteration eingeleitet wird.

4 Diskussion

Die Diskussion hat das Ziel unseren Ansatz mit existierenden Untersuchungen von LCDPs zu vergleichen, um Parallelen und Unterschiede aufzuzeigen. Dahingehend muss zunächst beantwortet werden, ob die konzeptionelle Modellierung und speziell die Entwicklung von Modellierungsmethoden, einen angemessenen Anwendungsbereich für Low-Code-Ansätze bilden, die traditionell im breiteren Software Engineering Kontext ansässig sind.

Die wachsende Nachfrage nach DSMMs spiegelt den zunehmenden Bedarf an Endbenutzeranwendungen wieder und unterstreicht zugleich die Notwendigkeit von anpassungsfähigen und agilen Modellierungsmethoden, um dem stetigen Wandel von domänenspezifischen Anforderungen gerecht zu werden (vgl. Abschn. 2.2). Dieser Bedarf wird noch durch einen Mangel an qualifizierten Methodenentwickler:innen verstärkt, der dem Defizit an qualifizierten Softwareentwickler:innen gleichkommt und somit die Dringlichkeit von integrativeren und zugänglicheren Entwicklungsplattformen betont. Es gilt dabei hervorzuheben, dass im Sinne des vorgestellten Low-Code-Ansatzes grundlegende Erfahrungen mit der verwendeten Sprache ADOSCRIPT erforderlich sind, um das volle Potenzial der Plattform auszuschöpfen. ADOxx ist dennoch so gestaltet, dass auch Nutzer:innen ohne Vorkenntnisse die vier Metakonzepte umsetzen können, wobei in einem solchen Fall Einschränkungen hinsichtlich der *Visualisierung* und speziell der *Umsetzung* bestehen würden (vgl. Abschn. 3.3). Dementsprechend bietet die ADOxx Plattform eine benutzerfreundliche Umgebung für die Entwicklung und Adaption von prototypischen Modellierungswerkzeugen durch *Citizen Developer* aus beliebigen Domänen, die keine tiefgreifenden Kenntnisse hinsichtlich Metamodellierung oder ADOSCRIPT besitzen. Darüber hinaus können Modellierungsexpert:innen angesprochen werden, die den Mehrwert einer gesteigerten Effizienz und Flexibilität bei der Methodenentwicklung nutzen wollen. Durch die Reduzierung des Codierungsaufwands und die Bereitstellung intuitiver Interaktionstools können Anwender:innen schneller und zielgerichteter anwendungsfertige Modellierungswerkzeuge erstellen und anpassen. In diesem Kontext ist es von großer Bedeutung, die Interoperabilität mit externen Diensten und Datenquellen zu ermöglichen, um den technischen und sich stetig wandelnden Herausforderungen, die beispielsweise mit der Integration von IoT, CPS und LLMs einhergehen, gerecht zu werden. Zusammenfassend kann also festgestellt werden, dass aktuelle Herausforderungen hinsichtlich der Entwicklung von DSMMs eine Reihe von Parallelen zu den ursprünglichen Beweggründen der Etablierung von LCDPs im Bereich des Software Engineering aufweisen (vgl. Abschn. 2.1).

Aufbauend auf der Angemessenheit von Low-Code-Ansätzen für die Realisierung von Modellierungsmethoden ist es relevant festzustellen, ob der in dieser Ar-

Tab. 4 Gegenüberstellung meistgenannter Eigenschaften von LCDPs und Low-Code-basierten Eigenschaften der ADOxx Plattform

Eigenschaft von LCDPs	Eigenschaften von ADOxx(LC)
Grafisches/Drag-and-Drop-basiertes Anwendungsdesign	Visuelles/Point-and-Click-basiertes Metamodell-Design Drag-and-Drop-basiertes Notation-Design
Vorgefertigte/Wiederverwendbare Komponenten (Formulare, Berichte, Dashboards, Referenzmodelle)	(Individualisierung) vorgefertigter Analysemechanismen
Interoperabilität mit Schnittstellen von externen Anwendungen (APIs)	Interoperabilität mit externen Diensten und Datenquellen
Integration externer Datenquellen	
Anpassung existierender Komponenten (Referenzmodelle, Datenmodelle, Prozesse)	Individualisierung vorgefertigter Analysemechanismen
Erweiterung durch neue Komponenten für das GUI Design	–
Bereitstellung in der Cloud	–
Bereitstellung auf lokaler Infrastruktur	Automatisierte Bereitstellung als Modellierungswerkzeug

beit vorgestellte Ansatz sowie die untersuchte Metamodellierungsplattform ADOxx die allgemeinen Anforderungen an eine LCDP erfüllen. Unter Berücksichtigung des theoretischen Hintergrunds formen die Architektur (vgl. Abschn. 2.1.1) und die Eigenschaften (vgl. Abschn. 2.1.2) einer LCDP die Basis für derartige Anforderungen. Der Ablauf des Low-Code-Ansatzes für die Entwicklung von Modellierungsmethoden (vgl. Abb. 1) folgt auf konzeptioneller Ebene der von da Cruz et al. (2021) vorgestellten Architektur, indem Metamodellierungsplattformen die jeweilige Entwicklungsumgebung darstellen, die den resultierenden Sourcecode durch externe Hilfswerkzeuge bereitstellen kann. Das AMME Vorgehensmodell stellt außerdem sicher, dass iterative Anpassungen von entwickelten Anwendungen gemäß sich ändernder Anforderungen vorgenommen werden können (Karagiannis 2015). Darüber hinaus weisen die auf verschiedenen Interaktionen basierenden Low-Code-Eigenschaften der ADOxx Plattform (vgl. Tab. 3) eine Vielzahl an Überschneidungen mit der Zusammentragung der meistgenannten Eigenschaften von LCDPs (vgl. Tab. 1) auf. Für eine Verdeutlichung der mehrfachen Überschneidung ist in Tab. 4 eine Gegenüberstellung der jeweiligen Eigenschaften festgehalten. In beiden Fällen sind die vier Ebenen einer generischen LCDP Architektur nach Sahay et al. (2020) durch die grundlegenden Eigenschaften abgedeckt (vgl. Abschn. 2.1.1).

Die vorausgegangenen Abschnitte der Diskussion präsentieren Antworten auf mögliche Defizite hinsichtlich der Angemessenheit von Low-Code-Ansätzen in der konzeptionellen Modellierung und den Low-Code-basierten Anforderungen an die ADOxx Plattform. Ein legitimer Kritikpunkt des vorgestellten Ansatzes, der dennoch verbleibt, ist das Argument, dass viele der präsentierten Eigenschaften der ADOxx Plattform sowie die externen Hilfswerkzeugen weder neu noch innovativ sind, da diese bereits seit längerer Zeit existieren. Dem kann die Schlussfolgerung von Bock und Frank (2021) im Zuge des Vergleichs von LCDPs entgegengestellt werden, wonach heutige Low-Code-Lösungen selten von Grund auf neu entwickelt wurden,

sondern häufiger Erweiterungen oder Neupositionierungen bestehender Produkte darstellen. Dahingehend werden insbesondere altbewährte Prinzipien der modellgesteuerten Entwicklung angeführt, deren erstmalige Verwendung bereits mehrere Jahrzehnte zurückliegt (Bock und Frank 2021; Lethbridge 2021). Der hier präsentierte Ansatz ist nicht erst kürzlich entwickelt worden, sondern basiert sowohl auf der iterativen Weiterentwicklung bestehender Funktionalitäten als auch auf der Integration neuer Erweiterungen. Es ist wichtig zu betonen, dass die Einbindung der vorgestellten, externen Komponenten (vgl. Abschn. 3.2) in die ADOxx-Plattform nicht vorgesehen ist. Diese hat den Hintergrund, dass eine kontinuierliche Erweiterung und Anpassung der Plattform durch interessierte Entwickler:innen ermöglicht werden soll, ohne durch die direkte Integration spezifischer Tools limitiert zu sein.

Die offene und erweiterbare Architektur von ADOxx ist in der Vergangenheit speziell durch Mitglieder der OMiLAB *Community of Practice* (Völz und Vaidian 2024) genutzt worden, die aktiv an der Weiterentwicklung der Plattform mitwirken. Die OMiLAB Community, bestehend aus Forscher:innen, Entwickler:innen und Praktiker:innen, trägt fortlaufend durch Erweiterungen auf unterschiedlichsten Ebenen dazu bei, dass sich wandelnde Anforderungen von Technologien und Nutzer:innen berücksichtigt und integriert werden können.[13] Zusätzlich ist die kontinuierliche Nutzung in zahlreichen Anwendungsszenarien durch die weitreichendere ADOxx Gemeinschaft und das daraus resultierende Feedback dafür verantwortlich, dass die existierenden Funktionalitäten und Erweiterungen stetig verbessert und evaluiert werden. Dieser iterative Entwicklungs- und Evaluationsprozess gewährleistet, dass die Komponenten nicht nur technisch ausgereift sind, sondern auch praktisch relevante Funktionen bieten, die den Anforderungen der Anwender:innen entsprechen.

5 Zusammenfassung

In diesem Beitrag wurde das transformative Potenzial von Low-Code-basierten Plattformen im Bereich der konzeptionellen Modellierung untersucht, wobei die Realisierung von domänenspezifischen Modellierungsmethoden und deren Bereitstellung als anwendungsfertige Modellierungswerkzeuge im Vordergrund stehen. Auf Basis des aktuellen Trends hinsichtlich Low-Code-Entwicklungsplattformen ist ein konzeptioneller Ansatz vorgestellt worden, der die Methodenrealisierung vereinfacht und den Entwicklungsprozess für Personen ohne einschlägige Fachkenntnisse zugänglich macht. Vier Metakonzepte, die grundlegende Anforderungen an eine anwendungsorientierte MM-DSL definieren, stehen im Mittelpunkt des Ansatzes.

Die Operationalisierung innerhalb der ADOxx Plattform demonstriert eine praktische Anwendung des Low-Code-Ansatzes für Metamodellierungsplattformen. Durch diese Integration wird die Fähigkeit einer MM-DSL verdeutlicht, grundlegende Metakonzepte auf einer konkreten Metamodellierungsplattform zu operationalisieren und damit ein agiles Vorgehen für die Entwicklung von Modellierungsmethoden anzubieten. Unser Beitrag besteht darin die Diskrepanz zwischen der

[13] Vor diesem Hintergrund ist das OMiLAB Code Repository etabliert worden, das eine Sammlung von Erweiterungen und Best Practices offen zur Verfügung stellt (siehe https://code.omilab.org/resources/).

technischen Komplexität der Methodenentwicklung und dem domänenspezifischen Fachwissen von *Citizen Developer* zu verringern, mit dem Ziel Anpassungsfähigkeit an die sich stetig entwickelnden Anforderungen auf dem Gebiet der konzeptionellen Modellierung zu fördern.

Die fortlaufende Entwicklung von LCDPs hat das Potenzial Ebenen der Zusammenarbeit, Innovation und Effizienz für die konzeptionelle Modellierung freizusetzen. Kollaborative Metamodellierungsumgebungen, in denen gemäß Prinzipien der Co-Kreation und kollektiven Intelligenz unterschiedliche Personen gemeinschaftlich zum Entwicklungsprozess von DSMMs beigetragen können, stellen einen vielversprechenden Ausgangspunkt zur Förderung eines interaktiven und dynamischen Ansatzes dar. Außerdem werden die Integration und Interoperabilität mit neuartigen Systemen der Künstlichen Intelligenz einen relevanten Bereich der Forschung einnehmen, um den Prozess der Methodenentwicklung weiter zu vereinfachen und zu automatisieren. Für die Zukunft gilt sicherzustellen, dass MM-DSLs und zugehörige Metamodellierungsplattformen benutzerfreundliche Integrationsmöglichkeiten für sowohl innovative Technologie als auch sich wandelnde Modellierungsstandards bieten.

Funding Open access funding provided by University of Vienna.

Literatur

Bock AC, Frank U (2021) In search of the essence of low-code: an exploratory study of seven development platforms. 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). IEEE, S 57–66 https://doi.org/10.1109/models-c53483.2021.00016

Bork D (2018) Metamodel-based analysis of domain-specific conceptual modeling methods. In: Buchmann RA, Karagiannis D, Kirikova M (Hrsg) The practice of enterprise modeling. Springer, Cham, S 172–187 https://doi.org/10.1007/978-3-030-02302-7_11

Buchmann RA, Karagiannis D (2016) Enriching linked data with semantics from domain-specific diagrammatic models. Bus Inf Syst Eng 5:341–353. https://doi.org/10.1007/s12599-016-0445-1

da Cruz MA, de Paula HT, Caputo BP, Mafra SB, Lorenz P, Rodrigues JJ (2021) OLP—A RESTful open low-code platform. Future Internet 13:249. https://doi.org/10.3390/fi13100249

Falcioni D, Woitsch R (2021) OLIVE, a model-aware microservice framework. In: Serral E, Stirna J, Ralyté J, Grabis J (Hrsg) The practice of enterprise modeling. Springer, Cham, S 90–99 https://doi.org/10.1007/978-3-030-91279-6_7

Frank U (2011) Multi-perspective enterprise modelling: Background and terminological foundation. Universität Duisburg-Essen, Institut für Informatik und Wirtschaftsinformatik (ICB), Essen

Frank U (2013) Domain-specific modeling languages: requirements analysis and design guidelines. In: Reinhartz-Berger I, Sturm A, Clark T, Cohen S, Bettin J (Hrsg) Domain engineering: product lines, languages, and conceptual models. Springer, Berlin Heidelberg, S 133–157 https://doi.org/10.1007/978-3-642-36654-3_6

Geiger M, Harrer S, Lenhard J, Wirtz G (2016) On the evolution of BPMN 2.0 support and implementation. 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE). IEEE, S 101–110 https://doi.org/10.1109/SOSE.2016.39

Handel MJ, Poltrock S (2011) Working around official applications: experiences from a large engineering project. Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work. ACM, S 309–312 https://doi.org/10.1145/1958824.1958870

Ibrahimi I, Moudilos D (2022) Towards model reuse in low-code development platforms based on knowledge graphs. Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings. ACM, S 826–836 https://doi.org/10.1145/3550356.3561570

Karagiannis D (2015) Agile modeling method engineering. In: Karanikolas N, Akoumianakis D, Nikolaidou M, Vergados D, Xeno M (Hrsg) Proceedings of the 19th Panhellenic conference on Informatics. Association for Computing Machinery, New York, S 5–10 https://doi.org/10.1145/2801948.2802040

Karagiannis D (2024) AdoScript: A DSL for developing conceptual modeling methods. In: Strecker S, Jung J (Hrsg) Informing Possible Future Worlds. Essays in Honour of Ulrich Frank. Logos, Berlin, S 287–308

Karagiannis D, Buchmann RA (2016) Linked open models: extending linked open data with conceptual model information. Inf Syst 56:174–197. https://doi.org/10.1016/j.is.2015.10.001

Karagiannis D, Kühn H (2002) Metamodelling platforms. In: Bauknecht K, Tjoa AM, Quirchmayr G (Hrsg) E-commerce and web technologies. Springer, Berlin, S 182 https://doi.org/10.1007/3-540-45705-4_19

Karagiannis D, Mayr HC, Mylopoulos J (Hrsg) (2016) Domain-specific conceptual modeling: concepts, methods and tools. Springer, Cham https://doi.org/10.1007/978-3-319-39417-6 (1 Ausg)

Karagiannis D, Burzynski P, Utz W, Buchmann RA (2019) A Metamodeling approach to support the engineering of modeling method requirements. 2019 IEEE 27th International Requirements Engineering Conference (RE). IEEE, S 199–210 https://doi.org/10.1109/re.2019.00030

Karagiannis D, Lee M, Hinkelmann K, Utz W (Hrsg) (2022) Domain-specific conceptual modeling: concepts, methods and ADOxx tools, 1. Aufl. Springer, Cham https://doi.org/10.1007/978-3-030-93547-4

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

Kühn H, Junginger S, Karagiannis D, Petersen C (1999) Metamodellierung im Geschäftsprozeßmanagement: Konzepte, Erfahrungen und Potentiale. In: Desel J, Pohl K, Schürr A (Hrsg) Teubner Reihe Wirtschaftsinformatik, Bd. 12923. Vieweg+Teubner, Wiesbaden, S 75–90 https://doi.org/10.1007/978-3-322-93104-7_5

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, Bd. 13036. Springer, Cham, S 202–212 https://doi.org/10.1007/978-3-030-89159-6_14

Mayr HC, Thalheim B (2021) The triptych of conceptual modeling. Softw Syst Model 20:7–24. https://doi.org/10.1007/s10270-020-00836-z

Mendes PN, Jakob M, Garcia-Silva A, Bizer C (2011) DBpedia spotlight: shedding light on the web of documents. Proceedings of the 7th International Conference on Semantic Systems. Association for Computing Machinery, New York, S 1–8 https://doi.org/10.1145/2063518.2063519

Michael J, Mayr HC (2015) Creating a domain specific modelling method for ambient assistance. 2015 Fifteenth International Conference on Advances in ICT for Emerging Regions (ICTer). IEEE, S 19–124 https://doi.org/10.1109/icter.2015.7377676

Muck C, Voelz A, Amlashi DM, Karagiannis D (2022) Citizens as developers and consumers of smart city services: a drone tour guide case. Companion Proceedings of the Web Conference 2022. Association for Computing Machinery, New York, S 1228–1236 https://doi.org/10.1145/3487553.3524848

Mylopoulos J (1992) Conceptual modelling and Telos. John Wiley & Sons, New York

de Oliveira RA, Cortes-Cornax M, Front A, Demeure A (2022) A low-code approach to support method engineering. Proceedings of the 25th International Conference on Model Driven Engineering Lan-

guages and Systems: Companion Proceedings. ACM, S 793–797 https://doi.org/10.1145/3550356.3561588
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
Platt R, Thompson N (2015) The evolution of UML. In: Khosrow-Pour MD (Hrsg) Encyclopedia of information science and technology, 3. Aufl. IGI Global, S 1931–1936 https://doi.org/10.4018/978-1-4666-5888-2.ch186
Rokis K, Kirikova M (2023) Exploring low-code development: a comprehensive literature review. In: Complex systems Informatics and modeling quarterly, S 68–86 https://doi.org/10.7250/csimq.2023-36.04
Rybiński K, Smialek M (2022) Beyond low-code development: marrying requirements models and knowledge representations. In: Ganzha M, Maciaszek L, Paprzycki M, Ślęzak D (Hrsg) Proceedings of the 17th conference on computer science and intelligence systems, Bd. 30. IEEE, S 919–928 https://doi.org/10.15439/2022f129
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). IEEE, S 171–178 https://doi.org/10.1109/SEAA51224.2020.00036
Voelz A, Amlashi DM, Lee M (2023) Semantic matching through knowledge graphs: a smart city case. In: Ruiz M, Soffer P (Hrsg) Advanced information systems engineering workshops. Springer, Cham, S 92–104
Völz A, Vaidian I (2024) Digital transformation through conceptual modeling: the NEMO summer school use case. In: Modellierung 2024. Gesellschaft für Informatik e. V., Bonn, S 139–156 https://doi.org/10.18420/modellierung2024_014