Plugin-Entwicklung: Kollaboratives Anforderungsmanagement

Dies ist ein Auszug aus dem Leitfaden 'Plugin-Entwicklung' der Plugin-Werkstatt von Moodle an Hochschulen e.V., der nach und nach in die Knowledge Base übertragen wird.

- Plugin-Entwicklung: Arten der Anforderungen
- Plugin-Entwicklung: Anforderungssammlung
- Plugin-Entwicklung: Anforderungsdokumentation
- Plugin-Entwicklung: Datenschutzanforderungen

Plugin-Entwicklung: Arten der Anforderungen

Was können Anforderungen im Sinne der Softwareentwicklung an mein Plugin sein?

Die präzise Definition von Anforderungen ist essentiell, um den Funktionsumfang eines Plugins einzugrenzen und zielgerichtete Entwicklung zu ermöglichen. Dabei wird bewusst auf technische Lösungsvorschläge verzichtet. Der Fokus liegt auf der Beschreibung, für wen das Plugin gedacht ist und was es letztlich leisten soll. Dazu lassen sich die Anforderungen nach ISO/IEC/IEEE 29148 wie folgt kategorisieren:

- **Funktionale Anforderungen**: Welche Aktionen können Nutzer*innen und Administrator*innen durchführen? Dazu zählen beispielsweise das Verwalten von Benutzerrollen und Rechten, das Erstellen, Bearbeiten und Löschen von Inhalten, Such- und Filterfunktionen, das Anlegen und Verwenden von Templates, Benachrichtigungen und Statusänderungen sowie Funktionen, die die Zusammenarbeit unterstützen, wie die gleichzeitige Bearbeitung und die Verfügbarkeit einer Versionshistorie.
- **Qualitätsanforderungen (Nicht-funktionale Anforderungen)**: Welche messbaren Qualitätskriterien sollen für das Plugin gelten (vgl. Kriterienliste kuratierte Liste)? Hierzu gehören Aspekte wie automatisierte Tests (PHP Unit Tests, Behat Tests), Performance, Skalierbarkeit, Unterstützung von Import/Export und LifeCycle, Usability, App-Unterstützung, responsives Design, Sicherheit und Stabilität, Dokumentation für Nutzende.
- **Rahmenbedingungen**: Woran muss sich das Plugin halten? Dazu zählen didaktische ebenso wie organisatorische, technische und rechtliche Vorgaben wie IT-Governance, Budgetgrenzen, feste Release-Termine, Systemkompatibilität, Barrierefreiheit, vorgesehene Schnittstellen, DSGVO-Konformität und Aufbewahrungsfristen für Daten.

Zur weiteren Eingrenzung des Plugin-Umfangs wird empfohlen, die MoSCoW-Priorisierung mit den Kategorien **Must have**, **Should have**, **Could have** und **Won’t have** zu verwenden. Insbesondere **Won’t have** dient als wichtiges Instrument zur klaren Scope-Eingrenzung und zum Erwartungsmanagement, indem bereits in der Konzeptionsphase bestimmte Anforderungen als *out of scope* definiert werden.

Um die Anforderungen systematisch auf Basis der ISO/IEC/IEEE 29148 zu erheben und anschließend mittels MoSCoW zu priorisieren, kommen geeignete Methoden zum Einsatz, die im nachfolgenden Kapitel näher erläutert werden.

Autor:innen: Stefan Dabrock (Moodle an Hochschulen e.V.), Philipp Dorok (Ruhr-Universität Bochum), Denise Materna-Stuttkewitz (TU Dortmund)

Plugin-Entwicklung: Anforderungssammlung

Wie sammle ich Anforderungen?

Anforderungen an das geplante Plugin können mit Hilfe der folgenden Methoden gesammelt werden:

Anforderungs-Workshop

Nutzt frühzeitig die Gelegenheit, die Akteure einzubeziehen, die mit dem Plugin arbeiten werden. Je nach Plugin-Idee können das Moodle-Administrator*innen und auch die Nutzer*innen des LMS sein. Ein entsprechender Workshop, bei dem diese Akteure ihre Bedürfnisse äußern können, bietet sich an, um die verschiedenen Perspektiven zu dokumentieren. Damit kann dann die Planung auf die Anforderungen der Akteure zugeschnitten werden.

Interviews & Clusterung

Eine gute Alternative zum Workshop ist es, mit den oben genannten Akteuren Interviews zu führen und so die Anforderungen an das Plugin zu identifizieren. Die Informationen aus den Interviews können dann zu Informationsclustern visuell aufbereitet werden.

Bestehende Lösungsmöglichkeiten analysieren und ggf. Lücken identifizieren

Ein wichtiger erster Schritt besteht stets darin die Plugins zu sichten, die es bereits gibt und sich mit der gleichen Problemlage befassen. So kann das Projektteam herausfinden, ob ein bestehendes Plugin in Kooperation mit den jeweiligen Maintainer*innen weiter entwickelt werden kann oder ob es sinnvoll ist, ein vollständig neues Plugin zu entwickeln. So können ggf. Synergien unter dem Stichwort „Das Rad braucht nicht neu erfunden werden“ genutzt werden.

Geeignete Quellen für die Recherche sind:

- Moodle Marketplace: Nachfolge des Plugin Directory
- Community Archive of Moodle Plugins (CAMP): Unabhängiges, Community-getragenes Plugin-Archiv als Alternative zum Moodle Marketplace
- Kurs Pluginwerkstatt im Vereins-Moodle
- Forum Aktiv genutzte und geplante Moodle-Plugins im Kurs „Moodle-Community“ im Vereins-Moodle

- Bereich Plugins in der Knowledge Base des Vereins
- Beispielanalyse Existing Solutions für den Quiz Archiver von Niels Gandraß

Workflow-Diagramm des zu lösenden Problems

Ein solches Diagramm bietet sich bei Plugins an, mit denen ein mehrschrittiger Arbeitsprozess durchgeführt wird. Ein gutes Beispiel für einen solchen Prozess ist die Buchung einer E-Prüfung, für die von der ersten Einreichung bis zum bestätigten Termin sowie der Prüfungsdurchführung eine Vielzahl miteinander verzahnter Schritte durch unterschiedliche Akteure (Buchende*r, Prüfende*r, Service-Team etc.) durchlaufen werden. Wenn im Vorfeld ein Diagramm mit allen Einzelschritten der einzelnen Akteure und ihrer Kommunikation entwickelt wird, dann hilft das dabei, die Anforderungen an das Plugin zu erkennen.

User Stories

User Stories sind ein gutes Mittel, um sich in die Lage der einzelnen Akteure zu versetzen, die mit dem fertigen Plugin arbeiten werden. Für jede*n Akteur*in – bei einem Buchungsprozess einer Prüfung beispielsweise die Personen des Serviceteams – kann aufgeschrieben werden, welche Aktionen die jeweiligen Personen durchführen können sollen und welche Erwartungen sie an Funktionen des Plugins haben. Hier geht es beispielsweise um die einzelnen Rechte (Schreib-, Lese- oder Administrationsrechte) der jeweiligen Akteure.

- Beispiel von User Stories für den Quiz Archiver von Niels Gandraß
- Beispiel für die Schritte der Anforderungsermittlung beim Quiz Archiver von Niels Gandraß (gilt bis Schritt „Formulation of User Stories“)

Autor:innen: Stefan Dabrock (Moodle an Hochschulen e.V.), Philipp Dorok (Ruhr-Universität Bochum), Denise Materna-Stuttkewitz (TU Dortmund)

Plugin-Entwicklung: Anforderungsdokumentation

Wie dokumentiere ich meine Anforderungen so, dass Entwickler*innen / Dienstleister*innen sie verstehen?

Die schriftlich dokumentierten User Stories bilden ein zentrales Element, um die Anforderungen an ein geplantes Plugin innerhalb der Projektgruppe festzuhalten. Gleichzeitig können sie auch externen Entwicklungsteams zur Verfügung gestellt werden, um diesen zu verdeutlichen, welche Ausgestaltung das Plugin haben soll. Da während des Entwicklungsprozesses neue Anforderungen auftauchen sowie gleichzeitig zuvor formulierte Aspekte der User Stories nicht mehr dem Bedarf entsprechen können, ist es sinnvoll die User Stories laufend zu prüfen und zu überarbeiten.

Parallel zur laufenden Überprüfung der User Stories ist es sinnvoll, die mit den unterschiedlichen Methoden eine Frage weiter oben gewonnenen Anforderungen in Issues zu übersetzen, also klar formulierten einzelnen Entwicklungsschritten, die zusammen das Gesamtbild des Plugins darstellen. Die Issues können dann in einem Planningboard im Kanban-Stil aufbereitet werden, wo sie entsprechend ihres jeweiligen Arbeitsstandes (Ausformulierte Überlegung, In Entwicklung, In der Testphase, etc.) die Position wechseln.

Dabei verweisen die im Planningboard nur grundsätzlich formulierten Issues per Link auf ihre ausführliche Darstellung in einem Github zum Plugin. Das Planningboard bietet die Möglichkeit, die einzelnen Issues konkreten Entwickler*innen zur Bearbeitung zuzuweisen.
Zentral sind noch sogenannte Mockups der einzelnen Funktionen, also bildliche Darstellungen der jeweiligen Nutzungs- oder Administrationsoberfläche wie sie im fertigen Plugin aussehen sollen. Sie sind in gewisser Weise ein Endpunkt der Anforderungen an das Plugin.

Sowohl bei einer rein internen Entwicklung und in jedem Fall bei einer Vergabe an externe Entwicklungsteams ist es sinnvoll, den Status Quo als Lastenheft zu dokumentieren. Es enthält eine grundsätzliche Beschreibung der Plugin-Funktionen und der Akteure, die damit arbeiten werden. Basierend auf den Issues wird zudem alles, was mit dem Plugin durchführbar sowie administrationsseitig einstellbar sein soll, mit Hilfe einer an Entwicklungsteams gerichteten Beschreibung formuliert. Dazu gehören auch die zuvor einzeln erstellten Mockups zur eindeutigen bildlichen Dokumentation. Auch wenn sich in einem dynamischen Prozess noch Änderungen am Plugin ergeben können, ist das Lastenheft die vorläufige Abschlussdokumentation, um den Menschen, die das Plugin entwickeln werden, einen klaren Auftrag zu geben.

Da die Entwicklung an einem Plugin grundsätzlich eine Änderungsdynamik besitzt, werden Issues, User Stories, Mockups sowie das Planningboard vom Entwicklungsteam laufend überprüft und angepasst.
Grundsätzliche schematische Darstellungen wie Organigramme oder Diagramme, die beispielsweise Prozesse und Rollen im Zusammenhang mit der Nutzung eines Plugins verdeutlichen, können ebenfalls eine sinnvolle Ergänzung sein.

Autor:innen: Stefan Dabrock (Moodle an Hochschulen e.V.), Philipp Dorok (Ruhr-Universität Bochum), Denise Materna-Stuttkewitz (TU Dortmund)

Plugin-Entwicklung: Datenschutzanforderungen

Habe ich Datenschutzanforderungen?

Grundsätzlich muss ein Plugin die Anforderungen erfüllen, die sich aus der Datenschutz-Grundverordnung (DSGVO) ergeben. Hierzu gehört, dass die Privacy API implementiert ist. Diese ermöglicht es, bei Auskunfts- oder Löschanfragen nach DSGVO die angefragten personenbezogenen Daten in menschenlesbarer Form zu exportieren oder diese zu löschen. Die Implementierung der Privacy API ist zudem eine Voraussetzung für eine Veröffentlichung des Plugins im Moodle Marketplace.

Gemäß dem Prinzip der Transparenz muss dokumentiert werden, welche Daten erzeugt, gespeichert oder übertragen werden. Dabei sind die Grundsätze der zweckgebundenen Datenverarbeitung und der Datenminimierung stets einzuhalten:

- Verbindungsdaten wie IP-Adresse, Browsertyp, Datum und Uhrzeit des Zugriffs
- Zugangsdaten wie Benutzername, Passwort in verschlüsselter Form, E-Mailadresse, Vorname, Nachname, …
- Inhaltsdaten wie hochgeladene Dateien, Forenbeiträge, Texte, …
- Prüfungsdaten im Rahmen von Prüfungen und Tests

Werden die Daten nicht allein lokal verarbeitet, sondern übertragen, ist der Standort der Datenverarbeitung zu dokumentieren, z.B. innerhalb oder außerhalb der EU.

Autor:innen: Stefan Dabrock (Moodle an Hochschulen e.V.), Philipp Dorok (Ruhr-Universität Bochum), Denise Materna-Stuttkewitz (TU Dortmund)