hochschule mannheim

Eine 3D Asset Pipeline: Von CAD über SAP zu Unity3D

Enes Ördek

Bachelor-Thesis

zur Erlangung des akademischen Grades Bachelor of Science (B.Sc.)

Studiengang Unternehmens- und Wirtschaftsinformatik

Fakultät für Informatik

Hochschule Mannheim

14.07.2021

Betreuer

Prof. Dr. Frank Dopatka, Hochschule Mannheim

Michael Spieß, SAP SE

Ördek, Enes:
Eine 3D Asset Pipeline: Von CAD über SAP zu Unity3D / Enes Ördek. –
Bachelor-Thesis, Mannheim: Hochschule Mannheim, 2021. 77 Seiten.

Ördek, Enes:
A 3D Asset Pipeline: From CAD via SAP into Unity3D / Enes Ördek. –
Bachelor Thesis, Mannheim: University of Applied Sciences Mannheim, 2021. 77 pages.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d. h. dass die Arbeit elektronisch gespeichert, in andere Formate konvertiert, auf den Servern der Hochschule Mannheim öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Mannheim, 14.07.2021

Enes Ördek

Abstract

Eine 3D Asset Pipeline: Von CAD über SAP zu Unity3D

SAP Kunden aus der diskreten Industrie nutzen immer häufiger Anwendungen, die auf der Unity Plattform basieren, um ihre 3D Produkte in Extended Reality für verschiende Anwendungsfälle (z.B. Marketing, Fertigung, Design) darzustellen. Kunden von SAP konvertieren ihre CAD Modelle in ein proprietäres Format (Visual Enterprise Datei, VDS), um diese mit Daten aus dem ERP anzureichern und performant in SAP Browseranwendungen anzuzeigen. Allerdings wird dieses Dateiformat nicht von Unity unterstützt, sodass bei der Konvertierung in ein unterstütztes Format (z.B. FBX) die ERP Daten verloren gehen. Um die Nutzung von Unity für Extended Reality Anwendungsfälle zu ermöglichen wurde das SAP XR Venture gegründet. Das SAP XR Venture hat das Smart Asset Konzept entwickelt, das prinzipiell alle 3D Plattformen unterstützt (bspw. Unreal Engine), um SAP Kunden das Teilen dieser 3D Assets, genannt Smart Assets, über Unternehmengrenzen hinweg zu ermöglichen. Das Erzeugen von Smart Assets ist bislang ein manueller und aufwändiger Prozess, bei dem Expertise für verschiedene Unity Versionen, Zielplattformen und Render Pipelines benötigt wird. Mithilfe einer 3D Asset Pipeline, die im Rahmen dieser Arbeit eigenständig entwickelt wurde, ist es zum ersten Mal möglich, den Prozess zum Erzeugen von Smart Assets zu automatisieren. Die erzeugten Assets können zur Laufzeit in alle 3D Anwendungen geladen werden, die mit Unity erzeugt wurden.

A 3D Asset Pipeline: From CAD via SAP into Unity3D

SAP customers in the discrete industry are increasingly using applications based on the Unity platform to display their 3D products in extended reality for various use cases (e.g. marketing, manufacturing, design). SAP customers convert their CAD models into a proprietary format (Visual Enterprise File, VDS) to enrich them with data from ERP and display them performantly in SAP browser applications. However, this file format is not supported by Unity, so when converting to a supported format (e.g. FBX), the ERP data is lost. To enable the use of Unity for extended reality use cases, the SAP XR Venture was formed. SAP XR Venture has developed the Smart Asset concept, which in principle supports all 3D platforms (e.g. Unreal Engine) to enable SAP customers to share these 3D assets, called Smart Assets,

across company boundaries. The creation of Smart Assets has been a manual and time-consuming process that requires expertise in different Unity versions, target platforms and render pipelines. Using a 3D asset pipeline that was developed as part of this work, it is possible for the first time to automate the process of generating Smart Assets. The generated assets can be loaded at runtime into all 3D applications created with Unity.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1. Motivation

Die digitale Transformation hat während der Corona Pandemie 2020/2021 eine unerwartete Entwicklung erlebt. Eine sich dynamisch ändernde Welt, aber auch das wandelnde gesellschaftliche Verhalten gefährden die Geschäftsmodelle etablierter Unternehmen [1]. Unternehmen, die ihre Geschäftsmodelle nicht an die neuen Gegebenheiten anpassen, sind immer weniger wettbewerbsfähig [2]. In diesem Kontext gewinnen Werkzeuge, die diese Transformation unterstützen eine große Bedeutung.

SAP SE (SAP) als global agierendes IT-Unternehmen sieht sich immer mehr in einem Umfeld, in dem Innovation eine strategisch essentielle Rolle spielt. "Innovation und Integration" sind in diesem Kontext die Leitsprüche, die SAP Vorstände immer wieder hervorheben. Erst im Januar 2021 hat Jürgen Müller (SAP CTO) und Thomas Saueressig (SAP Product Engineering) "Rise with SAP" angekündigt. Ein "Business Transformation As A Service" [3].

Unter dem Begriff Extended Reality (XR) werden werden alle Technologien zusammengefasst, die unsere Wahrnehmung umfassend beeinflussen. Die bekanntesten darunter sind Augmented Reality (AR), Mixed Reality (MR) und Virtual Reality (VR). Brunner und Weinberger haben Virtual Reality als zukunftsweisende Technologie um die digitale Transformation zu meistern, identifiziert [4]. XR besitzt Potenzial in verschiedenen Bereichen wie Produkt Produktentwicklung, Produktpräsentation und Fernbetreuung. Im Juni 2021 wurden die AR und MR Viewers für SAP Enterprise Product Development von Thomas Saueressig angekündigt [5].

Laut Bearingpoint setzt die XR Cloud neue Maßstäbe für die Umsetzung ganzheitlicher Transformationen [6].

1.2. Problem- und Fragestellung

Die Problemstellung dreht sich rund um 3D Assets. Die meisten 3D Assets lassen sich platformunabhängig lokal laden. Es werden sogenannte "Unity Prefabs" genutzt. Es handelt sich um ein proprietäres Textformat. Intern macht Unity also nichts anderes, als die Textdatei zu parsen und interpretieren. Das Nutzen von Prefabs um 3D Assets zu überliefern ist ein legitimer Weg. Doch man kommt in Hinsicht auf Performance sehr schnell an die Grenzen, falls man größere Modelle und Funktionalität nutzen möchte. Ebenfalls lassen sich keine neuen Inhalte mit Prefabs nach Ausliefern einer Unity Anwendung, laden. Bei Änderungen an 3D Modellen muss also die vollständige Anwendung neu kompiliert und ausgeliefert werden.

Kunden von SAP arbeiten meistens mit 3D CAD Modellen. Diese Modelle sind meistens sehr groß und umfangreich. Man will des weiteren Modelle modifizieren können und dann die aktualisierte Version erneut darstellen, ohne die vollständige Anwendung neu zu bauen. Der Ansatz über Unity Prefabs ist hier unbrauchbar. Zu diesem Zweck bietet Unity sogenannte "Unity Asset Bundles", die ein Binärformat darstellen. Sie müssen durch einen Kompiler laufen um erzeugt zu werden. Dieses Binärformat erlaubt es, auch sehr große Modelle performant in eine Unity-Szene zu laden und nutzen. Sie liegen in der Anwendung in Form einer Referenz vor und können jederzeit aktualisiert werden. Es können auch nachträglich neue Inhalte in Anwendungen ausgeliefert werden. Smart Assets nutzen dieses Konzept und erweitern sie mit Funktionalität (z.B. Animationen) und Daten (z.B. IOT Daten, Metadaten, ERP-Daten).

Das Ganze hat aber einen entscheidenden Nachteil: die 3D Objekte als "Unity Asset Bundles" sind abhängig von Unity Version, Platform und Render Pipeline. Es muss für jede Kombination also eine Binärdatei gebaut werden. Unity bringt monatlich eine neue Legacy Long-Termin-Support (LTS) Version und zweiwöchentlich eine neue Long-Termin-Support raus. Bei jeder neuen Version sind die alten Smart Assets in neueren Unity Versionen nicht mehr funktionsfähig oder es müssen neue gebaut werden. Bislang ist es ein manueller Prozess ein Smart Asset für mehr als eine

Unity-Version, Platform und Render-Pipeline zu erzeugen. Es erfordert Kenntnisse über Eigenheiten verschiedener Platformen, Unity Versionen und Render-Pipelines.

Neben all diesen Abhängigkeiten haben CAD Modelle ebenfalls ihre Eigenheiten: Es passiert, dass Objekte so komplex sind, dass sie nicht einmal im Binärformat geladen werden können. Für diese Zwecke hat das XR Venture "Best Practices" entwickelt. Dabei werden die Modelle vorher behandelt (z.B. Culler, Hierarchy Flattening), damit sie performant genutzt werden können. Auch diese müssen gegebenfalls auf die 3D Modelle vor dem Bauen der Binärdateien angewandt werden.

Wir führen Pioniersarbeit durch und die Machbarkeit ist offen. Die Fragestellung, mit der sich diese Arbeit beschäftigt ist, inwiefern wir eine 3D Asset Pipeline aufbauen können, in der diese Assets über eine Continous Integration (CI) Pipeline automatisiert erzeugt werden können, in der alle "Best Practices" und für verschiedene Unity-Versionen, Platformen und Render-Pipelines bereits eingebaut sind. Können wir so den Build-Prozess für 3D Assets (Smart Assets) skalieren?

1.3. Methodik

Diese Arbeit bedient sich der gestaltungsorientierten Forschungsmethode ("Design Science") [7]. Es wird analysiert, welche funktionalen und nicht-funktionalen Anforderungen eine Asset Pipeline haben muss. Dies wird durch Experteninterviews mit den Stakeholdern durchgeführt, mit dessen Hilfe Anforderungen und Ausblick beschrieben werden können. Zum Evaluieren der Lösung werden Testfälle beschrieben. Diese Testfälle nutzen die Asset Pipeline so, wie sie später auch von Kunden genutzt werden können.

1.4. Inhaltlicher Aufbau der Arbeit

Die Arbeit wird in sechs Kapitel aufgeteilt. Im Anschluss von der Einleitung folgt Kapitel 2, in der die Grundlagen verschiedener Technologien, Produkte und Dienste erläutert werden, die in dieser Arbeit relevant sind. Kapitel 3 analysiert die Umgebung, in der die Asset Pipeline eingesetzt werden soll und bescheibt, wo mögliche Schnittstellen bestehen. Kapitel 4 beschreibt die Implementierung. Die implemen-

tierte Lösung wird in Kapitel 5 evaluiert und Kapitel 6 schließt die Arbeit mit einem Ausblick und einem Fazit ab.

Kapitel 2

Grundlagen

Die Umgebung in der die Asset Pipeline arbeiten muss ist vielschichtig. Wir haben zum Einen die verschiedenen Cloud Umgebungen, zum anderen die verschiedenen Unity Anwendungen, in denen Inhalte produziert oder konsumiert werden. In der Grafik 2.1 ist ein Diagramm wie die Komponenten in der Softwarelandschaft miteinander in Beziehung stehen.

2.1. Extended Reality

Extended Reality (XR) ist ein Begriff, die alle Technologien umfasst, die unsere reale Welt immersiv erweitern und mit virtuellen Elementen kombinieren. Das X steht als Variable für alle gegenwärtigen und zukünftigen Technologien, die Wahrnehmungen von Raum beeinflussen. Im heutigen Kontext wird damit unter anderem Virtual Reality, Augmented Reality und Mixed Reality mit all ihren Abstufungen (z.B. Augmented Virtuality) umfasst.

2.1.1. Virtual Reality

Der Ausdruck "Virtual Reality" ist zum ersten Mal 1982 im Science-Fiction Roman "The Judas Mandala" von Damien Broderick verwendet worden [8, S. 14]. VR beschreibt Technologie, bei der man zumeinst mit Virtual Reality Headsets wie Oculus Quest eine virtuelle Umgebung wahrnehmen kann. Man kann mithilfe von Controllern, Körperbewegung, Handgesten, Sprachsteuerung usw. mit der virtuellen Umgebung interagieren. Virtual Reality hat bislang viel im Spieleumfeld großen

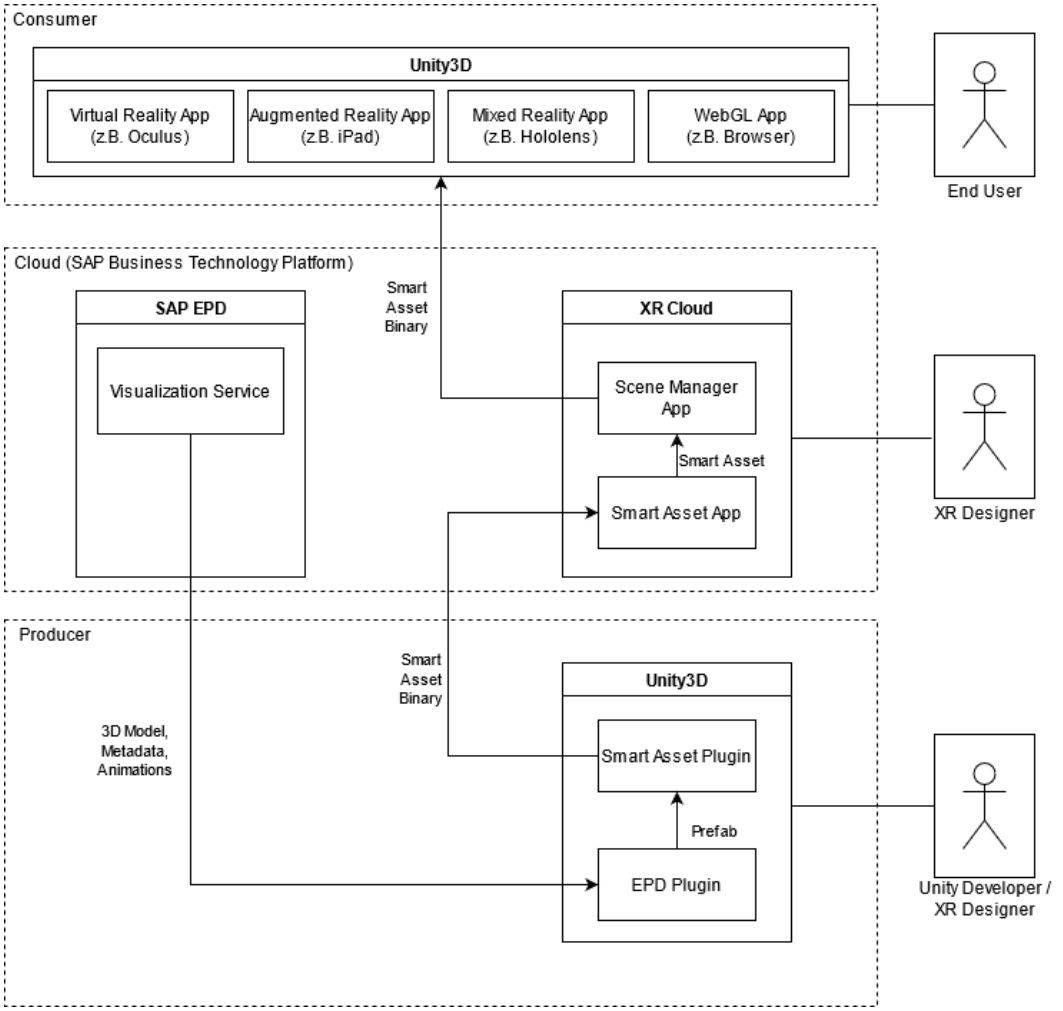


Abbildung 2.1.: Vereinfachte Architektur der Landschaft um die XR Cloud

Einfluss erlebt. Diese Technik wird aber auch immer mehr im unternehmerischen Umfeld genutzt, wie beispielsweise für VR-Trainings.

In 2.2 sieht man ein Screenshot aus einer VR Trainingszene.

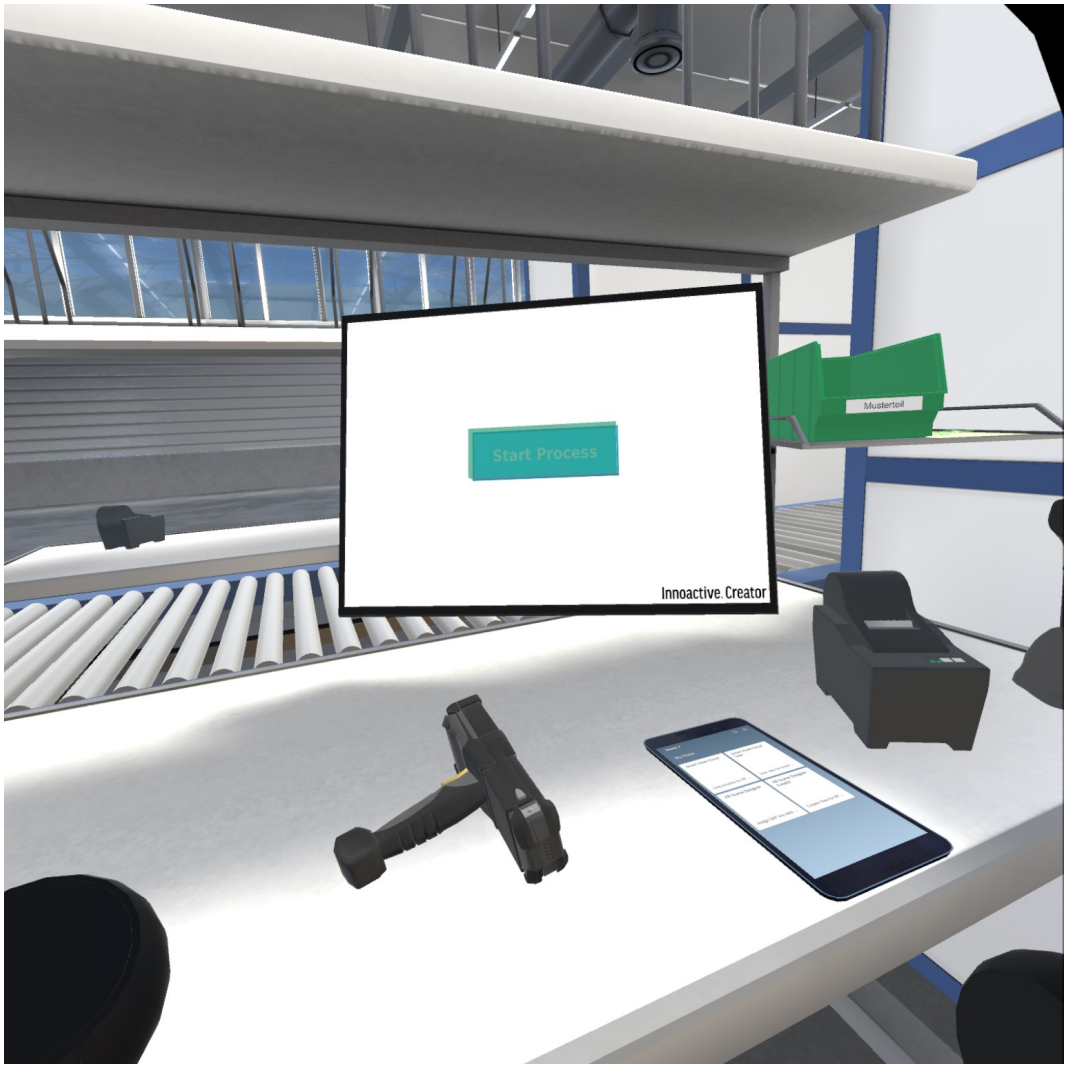


Abbildung 2.2.: Screenshot aus einer VR Trainingsszene

2.1.2. Augmented Reality

Augmented Reality (AR) ist laut Azuma [9, S. 356] eine Variante von VR. Anders als bei VR sieht der Nutzer bei AR seine physische Umgebung. Es sind virtuelle Objekte in der phyisischen Umgebung dargestellt. AR ergänzt die Realität, statt sie vollständig zu ersetzen.

Typischerweise werden hierfür Smartphones oder Tablets genutzt um die Realität mit weiteren Elementen zu erweitern. Im Kontext dieser Arbeit ist mit AR Anwendungen gleichzusetzen mit XR Anwendungen auf mobilen Bildschirmen, wie Smartphone oder Tablets.

In 2.3 ist solch eine AR Anwendung zu sehen. Im Hintergrund befindet sich ein physisches Objekt. Der Benutzer sieht in der iPad-App das Innere des Objektes und die zugehörigen Metadaten der Teile.

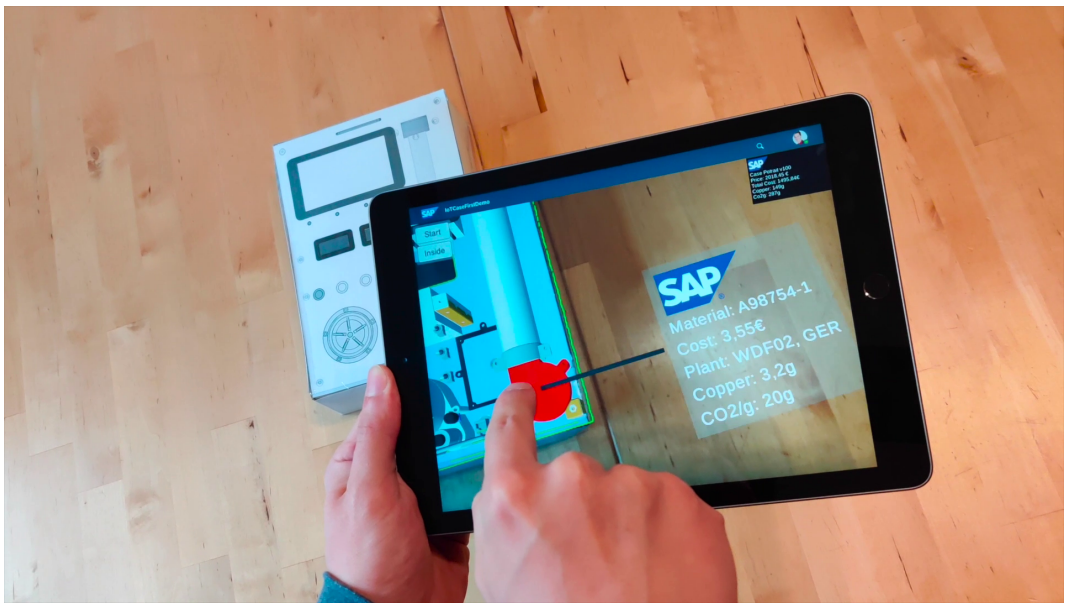


Abbildung 2.3.: AR Anwendung des XR Venture

2.1.3. Mixed Reality

Mixed Reality (MR) besitzt keine eindeutige Defintion und die Bedeutung ist vom Kontext abhängig [10, S. 12]. Im Kontext von SAP ist es mit Anwendungen für Geräte wie Hololens gleich zu setzen. Es ist eine Mischung aus physischer und digitaler Welt [11]. Anders als bei AR existiert eine größere Anzahl von Interaktionsmöglichkeiten, die als Schnittstelle zwischen den beiden Welten dienen (z.B. Handgesten, Sprachgesten, Blickgesten). Prominent wurde MR besonders durch Microsoft Hololens und dem Mixed Reality Toolkit. In der Darstellung 2.4 sieht man die Funktionsweise einer MR App. Die Nutzer der App tragen die Hololens und sehen aus ihrer Sicht das, was in der Darstellung sichtbar ist.

Abbildung 2.4.: Mixed Reality Anwendung des SAP Mixed Reality Solutions Team [12]

2.2. Unity3D

Unity3D (oft auch nur "Unity") ist eine Platform (auch "Game-Engine") um 3D Anwendungen zu entwickeln und wird vom gleichnamigen Unternehmen entwickelt. Sie ist ursprünglich zur Entwicklung von Spielen gedacht, man kann damit aber alle möglichen 3D Anwendungen erstellen. So auch XR Anwendungen und damit jede Art von Virtual Reality, Augmented Reality und Mixed Reality Anwendungen.

Unity umfasst eine Vielzahl von Werkzeugen. Neben einer umfangreichen graphischen Oberfläche zum Aufbauen der Spielelogik [13, S. 3], wird für Unity An-

wendungen Dotnet als zugrundeliegende Entwicklungsplatform genutzt. Entwickler schreiben Logik in der Programmiersprache C#, diese Logik wird mit den graphischen Elementen verbunden und es entsteht so eine 3D Anwendung.

Mit Unity wird ein Unity Projekt aufgebaut, welches alle möglichen Dateien, Metadaten und Abhängigkeiten zum Bauen einer Unity Anwendung enthält. Ein Unity Projekt ist platformunabhängig. Die selbe Anwendungslogik kann, abgesehen von wenigen Besonderheiten einer jeweiligen Platform, für jede beliebige Platform gebaut werden. So kann beispielsweise mit einem einzigen Unity Projekt viele verschiedene Benutzeranwendungen für Android, iOS, WebGL, Windows usw. gebaut werden (siehe 2.2.2).

Dies wird auch als besonderes Merkmal für Entwicklungsplatformen wie Unity angesehen: Hardwarehersteller möchten den Entwicklern einen möglichst einfachen Eintritt in die Entwicklung ihrer Hardware gewähren. Dies ist einfach dadurch erzielt, dass man als Hardwarehersteller eine Unterstützung für eine Entwicklungsplattform wie Unity entwickelt. Entwickler dagegen möchten sich die Einarbeitung in eine neue Plattform ersparen und bevorzugen ihnen bekannte Werkzeuge. So kommt ein Ökosystem zustande.

Es existieren vergleichbare Entwicklungsplattformen. Die prominenteste ist die Unreal Engine. Unity hebt sich im Vergleich zur Unreal Engine mit dem Marktanteil hervor. Dies wird auch in der Grafik 2.5 deutlich. Während Unity 2020 ein Marktanteil unter Entwicklern bei 10,9% besaß, lag dieser Anteil bei der Unreal Engine bei 3,3%. Es ist auch oftmals die gute Community, weshalb sich Entwickler für Spieleplattformen entscheiden [13, S. 3].

Unternehmen wie SAP haben dies erkannt und investieren bereits jetzt in eine Infrastruktur und Werkzeuge, um ihren Kunden den Zugang in dieses Ökosystem zu bieten.

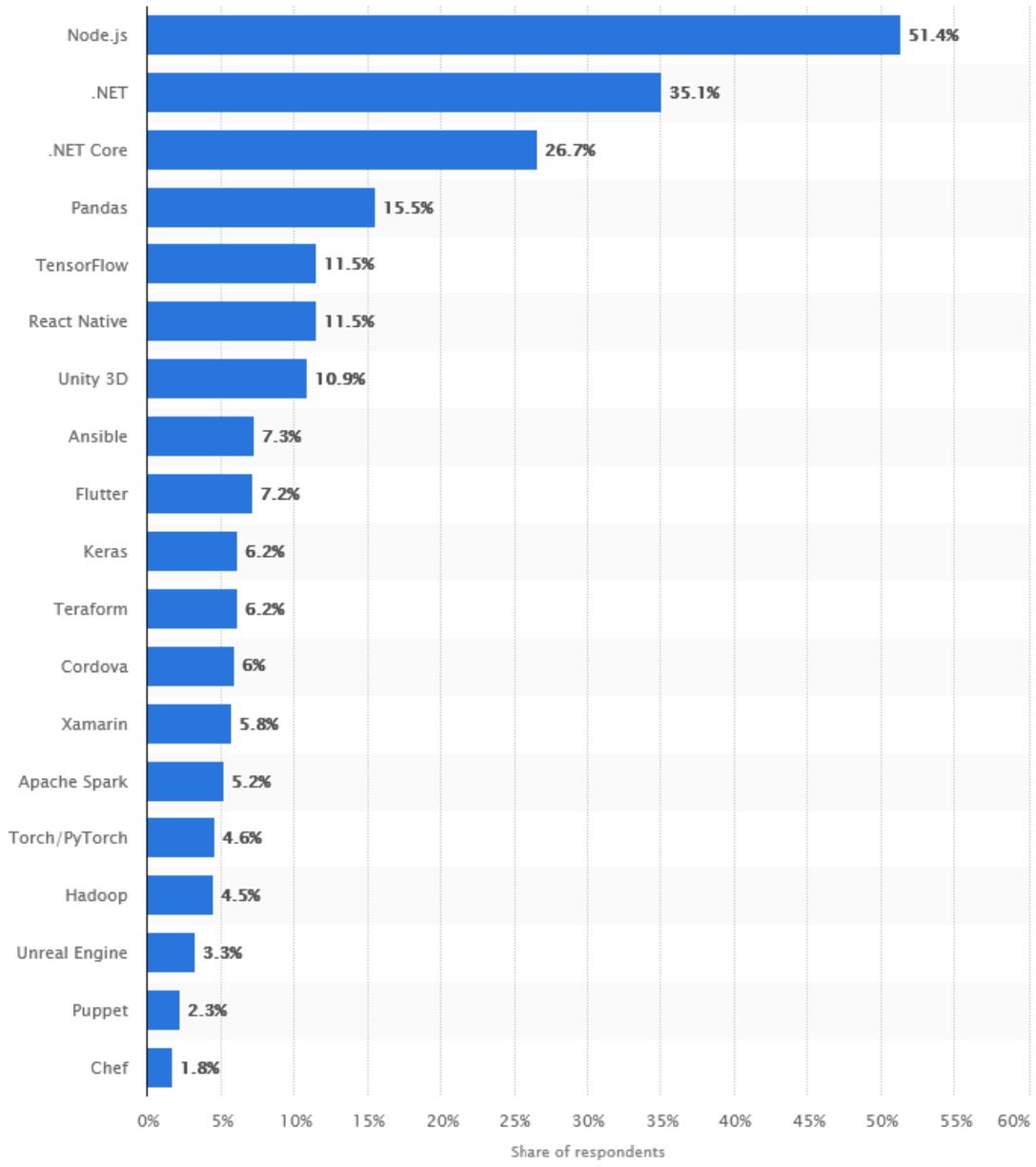


Abbildung 2.5.: Statista, am Meisten genutzte Bibliotheken, Frameworks und Werkzeuge unter Entwicklern, weltweit, 2020 [14]

2.2.1. Unity Version

Es existieren verschiedene Versionsstränge, die von Unity parallel gepflegt werden. Es erscheint täglich ein neues Release (Alpha/Beta), die jedoch instabil sind und bei dem kein Support angeboten wird. Besonders beliebt sind daher die Long-Term-Support (LTS) Versionen. Bei diesen wird garantiert, dass die Entwicklungsumgebung bei jedem Update auf die nächste Version des selben Stranges stabil bleibt. So existiert zum aktuellen Zeitpunkt ist 2019.4.x die Legacy-LTS, mit einer Unterstützung bis Frühling 2022 und monatlichen Releases, und 2020.3.x die LTS Version mit einer Unterstützung bis März 2023 und zweiwöchigen Releases [15].

2.2.2. Zielplattform

In der nativen Entwicklung von Anwendungen ist man oft mit hohen Kosten und hohem Aufwand konfrontiert, falls man eine bestimmte Anwendung auf mehrere Plattformen umsetzen möchte. Mit Unity spart man sich diesen Aufwand. Laut eigenen Angaben unterstützt Unity mehr als 20 Plattformen [16]. Dies wird unter dem Motto "Build once, deploy anywhere" vertrieben und offenbart den Vorteil, den Plattformen wie Unity anbieten. Unity Projekte müssen mithilfe von Unity Modulen für die jeweilige Plattformen gebaut werden. Der Entwickler kann in Unity Hub zusammen mit der Version installieren und das Nutzen einer weiteren Zielplattform ist nicht mehr, als das Installieren eines neuen Moduls. Beim Bauen einer Anwendung entsteht je nach Zielplattform unterschiedliche Build-Artefakte. Dies kann im Falle von Android eine einfache Binärdatei (APK) zum Installieren sein, im Falle von iOS jedoch auch ein neues vollständiges XCode-Projekt.

2.2.3. Unity Scene

Eine Szene in Unity kann mehreres bedeuten. Zum Einen wird damit der Inhalt gemeint, bei dem mehrere Unity GameObjects miteinander interagieren. Zum Anderen wird damit aber auch das Software Artefakt gemeint, in der eine solche Szene persistiert wird. Unity Szenen sind nicht zu verwechseln mit XR Cloud Szenen (siehe 2.5.2).

2.2.4. GameObject

GameObject ist eine Klasse in Unity, von der alle Objekte, die in eine Szene geladen werden können, erben [17]. Zu diesen Objekten zählen Modelle, Kameras, Lichtobjekte usw. Auch Prefabs als persistierte Objekte, die nicht unbedingt in einer Szene sind, sind von der Klasse GameObject. Ein GameObject kann mehrere Komponenten (Klasse "Component") enthalten, die ein Objekt beschreiben. In einer Transform-Komponente wird beispielsweise die Position, Skalierung und Rotation eines Objekts beschrieben.

2.2.5. Render Pipeline

Eine Render Pipeline führt eine Menge an Operationen aus, um Inhalte darzustellen. Zu diesen Operationen gehören beispielsweise folgende.

1. Culling: Nicht sichtbare Objekte werden ausgeblendet um Rechenleistung zu sparen.
2. Rendering: Wie Objekte zu sehen sind.
3. Post Processing: Das Anwenden von weiteren Effekten und Filtern auf einer Darstellung.

Es existieren die folgenden vier Render Pipelines, die von Unity aktuell unterstützt werden [18].

1. Built-In Render Pipeline (Built-In RP): Das ist die Render Pipeline, die standartmäßig in Unity Projekten angewendet wird. Der Nachteil dieser Pipeline ist, dass sie Ressourcenintensiv ist, und bedingt für mobile Anwendungen geeignet ist.
2. Universal Render Pipeline (URP): Bei URP handelt es sich um eine voreingestellte Scriptable Render Pipeline. Immer mehr ersetzt URP die Built-In RP. Sie ist sehr gut für mobile Anwendungen geeignet.
3. High Definition Render Pipeline (HDRP): Unity findet sich oft der Kritik ausgesetzt, dass sie von der grafischen Qualität her der Unreal Engine unterlegen ist. Um diese Kritik anzugehen hat Unity diese Render Pipeline entwickelt. Es handelt sich ähnlich wie bei URP um eine voreingestellte Scriptable Ren-

der Pipeline, mit der Nutzer sehr schnell ein vorkonfiguriertes Unity Projekt für hohe grafische Anforderungen starten können.

4. Scriptable Render Pipeline (SRP): Die Scriptable Render Pipeline ist die frei konfigurierbare Render Pipeline, die besonders für fortgeschrittene Nutzer gedacht ist.

2.2.6. Unity Artefakte

Unity nutzt eine Zahl an Artefakte, um Inhalte zu übermitteln. Zwei Arten dieser Artefakte sind für die Asset Pipeline relevant.

Asset Bundles

Asset Bundles sind ein proprietäres Binärformat von Unity, mit der sich Inhalte an Zielanwendungen übermitteln lassen. Sie können Modelle, Texturen, Meshes, Prefabs, Audiodateien und sogar ganze Szenen enthalten [19]. Da es sich um ein Binärformat handelt, können diese Inhalte auch performant in eine 3D Szene geladen werden. Asset Bundles wurden ursprünglich entwickelt, um herunterladbare Inhalte (Downloadable Content, DLC) anbieten zu können. Da die Anwendung, die diese herunterladbaren Inhalte nutzen können soll meistens im Vornherein klar ist, sind Asset Bundles meistens für eine bekannte Kombination aus Unity Version, Render Pipeline und Target Platform gebaut. Man kann vorhandene Unity Anwendungen mit neuen Inhalten erweitern, auch nachdem sie auf Zielgeräten bereits installiert sind.

Im XR Venture wurde die Asset Bundle Technik zunutze gemacht, um generische Anwendungen zu erzeugen, in der alle Assets als Asset Bundles kommen. Dieses Konzept wird als Smart Asset Konzept bezeichnet. Anders als Unity Asset Bundles richten sich Smart Assets an beliebige (im Vornherein unbekannte) Unity Endanwendungen. So sind Endbenutzeranwendungen leichtgewichtiger, man kann vorhandene Inhalte jederzeit aktualisieren und plattformübergreifend neue Inhalte anbieten. Mehr zu Smart Assets in 2.4.

Prefabs

Prefabs sind die persistierte Form von 3D Objekten. Sie können beliebig oft instanziiert werden. Es handelt sich dabei technisch Gesehen um ein Textformat, bei dem mithilfe einer Beschreibungssprache Unity Komponenten (3D Daten, Positionsdaten, Parameter, Felder usw.) miteinander über Referenzen verbunden werden. Im Vergleich zu Asset Bundles, die ein Binärformat darstellen, müssen Prefabs zum Zeitpunkt der Entwicklung im Projekt existent sein, damit diese genutzt werden können. Ebenfalls können Prefabs, anders als Asset Bundles, nicht ohne weiteres von einem Unity Projekt in das andere Unity Projekt "kopiert"werden. Da ein Prefab nur Referenzen enthält, müssen auch alle abhängigen Referenzen mit kopiert werden. Hierfür nutzt man oft ein Package.

Packages

Ein Unity Package ist ein Archiv. Es kann 3D Daten, Prefabs, Szenen und alle möglichen Dateien beinhalten, die in einem Unity Projekt existieren können. Packages werden dazu genutzt, um Inhalte von einem Projekt in ein anderes Projekt zu übertragen [20].

Es existiert ein Package Manager, mit dem Package Abhängigkeiten aus verschiedenen Quellen installiert und genutzt werden können. Unity liefert standartmäßig Packages aus. Packages können also zum Einen manuell entpackt und genutzt werden, zum Anderen über den Package Manager automatisiert heruntergeladen und genutzt werden [21].

2.2.7. Edit-Mode und Play-Mode

Unity besitzt zwei Modi, mit der Anwendungen betrieben werden können, der Edit-Mode (auch Unity Editor) und der Play-Mode (auch Unity Runtime). Der Edit-Mode wird beim Entwickeln der Anwendung genutzt. Hier werden die verschiedenen Gameobjekte in einer Szene zusammen gebracht und miteinander in Verbindung gebracht. Das so resultierende Ergebnis kann anschließend getestet werden, indem auf den Play-Mode gewechselt wird.

Im Play-Mode ist die Spielkamer aktiv und es wird die Ansicht dargestellt, die der Endanwender sieht, wenn er die Anwendung aufruft. Technisch gesehen wird in

beiden Modi die 3D Information gerendert, sodass in beiden das Gleiche zu sehen ist. Was sich jedoch unterscheidet ist, dass im Play-Mode zusätzlich die Scripte/-Komponenten ausgeführt werden, die an den Gameobjekten angeheftet sind [22].

2.2.8. Scripts / Monobehaviour

Alle ausführbaren Scripte in Unity erben von der Klasse Monobehaviour. Wird in den Play Mode gewechselt oder eine Unity Endanwendung ausgeführt, so führt die Laufzeitumgebung die Scripte dieser Klasse aus, die in der aktiven Szene an aktiven Gameobjekten als Komponente angehängt sind. Die Ausführung der Scripte folgt einer Ausführungsreihenfolge [23].

2.2.9. Unity Lizenz

Die Unity Lizenz ist für Studierende kostenlos. Für kleine Unternehmen mit einem Jahresumsatz von weniger als 200.000 $ kostet die Unity Lizenz 399 $ pro Entwickler. Ab diesesm Jahresumsatz kostet sie 1.800 $ pro Jahr pro Entwickler. Im Endbenutzer Lizenzvertrag von Unity wird definiert, wie es um die geistigen Eigentumsrechte bei Unity und deren Build Artefakten steht [24].

> 4. Intellectual Property Rights
>
> 4.1 Unity's Ownership
>
> The Site, Software, Developer Services, Communities and Website Content (as that term is defined in the Site and Communities Additional Terms) are protected by copyright, trademark, and other laws of the United States and foreign countries. Except as expressly provided in the Agreement, Unity and its licensors exclusively own all right, title and interest in and to the Services, including all associated intellectual property rights. You will not remove, alter or obscure any copyright, trademark, service mark or other proprietary rights notices incorporated in or accompanying the Services.
>
> 4.2 Your Content
>
> As between you and Unity, **you own all right, title and interest (including, all intellectual property rights) in and to the content you**

> **create using the Software**, Developer Services and/or any content you post to the Site or in the Communities (collectively, "Your Content") (other than any components of the Software contained therein or used in connection therewith).

Mithilfe einer Unity Lizenz kann man den Unity Editor nutzen und Unity Projekte erstellen. Hat man eine Unity Anwendung entwickelt, so wird die Anwendung als ausführbares Build Artefakt exportiert. Dieses Build Artefakt kann kostenlos verteilt und ausgeführt werden. Gewöhnlich sind es diese Build Artefakte, die beispielsweise Spielestudios in Appstores veröffentlichen, damit Kunden sie installieren und nutzen können. Auch Unity Packages und Unity Asset Bundles (und damit Smart Assets) sind Unity Build Artefakte.

2.3. SAP Enterprise Product Development

Enterprise Product Development (EPD) ist ein SAP Produkt, bei dem Kunden im Entwicklungsprozess ihrer Produkte unterstützt werden [25]. Bei EPD kann man verschiedene Services dazubuchen. Eine dieser Services ist der Visualization Service. Im Rahmen dieser Arbeit wird EPD als Bezeichnung für den SAP EPD Visualization Service genutzt.

Der Visualization Service ist SAPs eigene Grafikengine, mit der 3D Modelle in der Cloud erzeugt, konsumiert und bearbeitet werden. Eine Funktionalität dieses Services sind beispielsweise Animationen. Mithilfe von Animationen können Arbeitsanweisungen definiert und dargestellt werden. Der Geschäftsnutzer kann so zum Beispiel Arbeitsanweisungen einsehen, aber auch zu Trainingszwecken die Wartung von bestimmten Produkten oder Geräten einsehen. Eine weitere Funktionalität ist das Verbinden der 3D Modelle mit Daten verschiedener Quellen. Im SAP Ökosystem entstehen allerlei Daten. Modelle können zum einen bereits über CAD Daten mit geliefert bekommen, es können aber auch Daten aus anderen Produkten und System anfallen, die dank einer direkten Anbindung an das jeweilige System auch durchgehend aktuell sind.

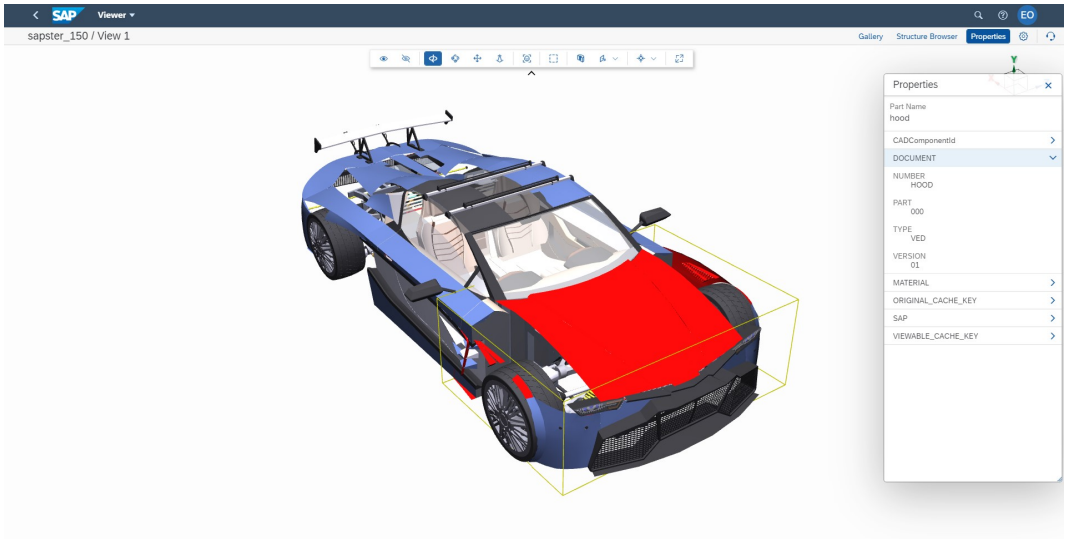


Abbildung 2.6.: Visualisierung des "Sapster"-Modells in SAP EPD Cloud mit Metadaten

2.4. Smart Asset

Ein Smart Asset enthält Unity Softwareartefakte (Asset Bundles und Unity Packages), die in der XR Cloud für Mehrfacheinsatz in XR Anwendungen konfiguriert werden können. Smart Assets sind „smart“ weil sie sowohl die 3D Informationen zum Darstellen der Objekte, als auch Parameter, Programmcode und Metadaten enthalten. In 2.7 ist das Datenmodell der XR Cloud dargestellt. Pro Smart Asset Version enthält ein Smart Asset viele Binaries, die technisch gesehen Unity Asset Bundles sind. Smart Asset Packages sind technisch gleich zu setzen mit Unity Packages.

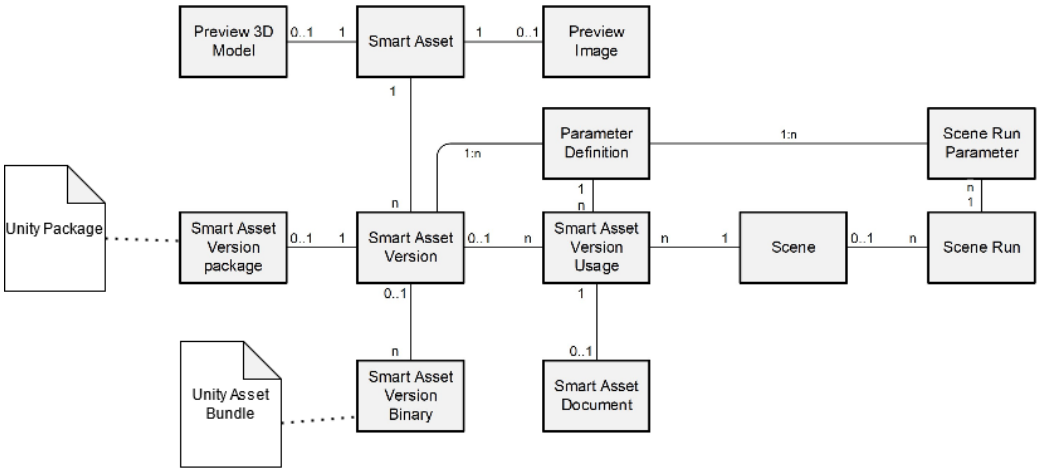


Abbildung 2.7.: Datenmodell der XR Cloud [26]

2.5. XR Cloud

Die XR Cloud ist ein Cloud Service, welches vom SAP XR Venture entwickelt wird. Das Entwickeln von 3D Anwendungen erfordert Expertise in vielen verschiedenen Bereichen. Es werden Entwickler gebraucht, Grafikdesigner, UX-Designer, Process Designer usw. Dadurch ist das Erzeugen von 3D Anwendungen teuer und aufwendig. Die Aufgabe der XR Cloud ist es, das Aufsetzen und Nutzen von 3D Anwendungen zu skalieren. Mit Unity werden generische Unity Anwendungen gebaut. Inhalte dieser Anwendung werden über die Web Oberfläche der XR Cloud bestimmt. So können Kunden Anwendungen anpassen, ohne dass weitere Entwickler beauftragt werden müssen. Die Corporate Identitiy, 3D Objekte, Prozesse und Daten können ohne Entwicklerkenntnisse beeinflusst werden.

Die XR Cloud wird auf der SAP Business Technology Platform (BTP) betrieben, einer SAP Plattformlösung. SAP BTP nutzt Hyperscaler wie Google Cloud, Amazon AWS oder Microsoft Azure als Laufzeitumgebung. Die XR Cloud und alle ihre Apps haben dadurch eine verteilte Infrastruktur.

2.5.1. Smart Assets

Die Smart Assets App ist eine App, in der Smart Assets aufgelistet sind. Über die Smart Asset App können Kunden neue Smart Assets anlegen, diesen Informationen ergänzen und Versionen hinzufügen. In diesen Versionen können 3D Inhalte als Package hinzugefügt werden. Diese Packages enthalten alle nötigen Dateien, um ein abgeschlossenes Asset Bundle zu erzeugen. Die erzeugen Asset Bundles können wieder als Smart Asset Binary in die Smart Asset Versionen strukturiert hochgeladen werden.

2.5.2. XR Scenes

Die XR Scenes App nutzt Inhalte aus der Smart Assets App, um zu bestimmen, welche Smart Assets in einer XR Cloud Szene hinzugefügt werden sollen. Eine XR Cloud Szene besitzt eine eindeutige Kennung und kann in Unity Anwendungen referenziert werden. Nutzer können so den Inhalt einer Unity Anwendung beeinflussen. In der Scenes App können zu den einzelnen Smart Assets Parameter zugeordnet werden. Diese Parameter können manuell hinzugefügt, oder aus SAP Systemen

stammen. Auf diese Weise findet beispielsweise die Verbindung von 3D Objekten mit echten Daten aus SAP ERP Systemen statt. Dies ist dann entscheidend, wenn man beispielsweise echte SAP Prozesse in XR abbilden möchte (z.B. als Training oder Simulation). Eine XR Cloud Szene richtet sich exakt für eine bestimmte Unity Version, Render Pipeline und Zielplattform. Das heißt, dass im Vornherein festgelegt werden muss, in was für einer Umgebung eine XR Cloud Szene genutzt wird.

2.5.3. XR Cloud Configuration

In der XR Cloud Configuration App kann die XR Cloud eingestellt werden. Hier werden die Zugänge zu anderen Systemen festgelegt.

2.5.4. Smart Asset Test Room

Der Smart Asset Test Room ist eine Unity Anwendung, die für den Browser gebaut wurde. Die zugrundeliegende Technologie ist WebGL. Es handelt sich um eine vollwertige, im Browser lauffähige Unity Anwendung, die das Smart Asset Plugin nutzt, um eine Szene aus der Scene Manager App zu laden. Mithilfe dieser Anwendung kann geprüft werden, ob ein Smart Asset so funktioniert, wie es beabsichtigt ist. Es können die Parameter und Animationen eingesehen werden.

2.6. Unity Integration Toolkit

Unter dem Begriff Unity Integration Toolkit (UIT) wird eine Reihe von Anwendungen bezeichnet, die zur Integration von SAP Anwendungen und Dienste in Unity dienen.

2.6.1. Smart Asset Plugin

Das Smart Asset Plugin integriert die Kernfunktionalität der XR Cloud in Unity. Das Plugin besteht aus zwei Teilen. Dem Smart Asset Loader und dem Smart Asset Bundler. Der Loader ist dafür zuständig, Smart Assets und XR Cloud Szenen über

Referenzen in eine Unity Szene zu laden. Hierfür werden Programmierschnittstellen angeboten, die das ermöglichen.

Mithilfe des Bundlers können Smart Assets aus Prefabs erzeugt und hochgeladen werden. Dies wird über eine grafische Oberfläche gesteuert.

2.6.2. SAP Enterprise Product Development Unity Plugin

Das SAP Enterprise Product Development Unity Plugin (auch SAP EPD Plugin) ist ein Plugin, um 3D Inhalte, Metadaten und Animationen aus SAP EPD in Unity Editor und Runtime zu laden. Das Plugin besitzt keine grafische Oberfläche, sondern Programmierschnittstellen zum Nutzen verschiedener Funktionalitäten. Neben den Kern-EPD Funktionalitäten (wie Animationen und Metadaten) wurden hier auch verschiedene Best-Practices als Funktionalität ergänzt, die über Parameter gesteuert werden können. Diese Best-Practices dienen dazu, beispielsweise komplexere Modelle darstellen zu können durch beispielsweise Culling, Hierarchy Flattening etc.

2.7. Azure Devops

Azure Devops ist ein Cloud Dienst von Microsoft Azure. Azure Devops ist vergleichbar mit Open-Source Automatisierungssoftware wie Jenkins. Mit Azure Devops können Prozesse zum Erzeugen von Software Artefakten automatisiert werden.

Kapitel 3

Analyse

3.1. Motivation

Es wurde zur Sapphire 2021 die Viewer Apps für iOS, Android und Hololens angekündigt [5]. Diese Apps werden ebenfalls vom XR Venture entwickelt und stellen SAP EPD Modelle mit ihren Animationen und Metadaten dar. Bisher werden Modelle aus dem SAP EPD direkt aus der SAP EPD Cloud in eine Unity Anwendung geladen. Dies nutzt aber nicht die Vorteile der Smart Assets (siehe 3.1.1). Die XR Cloud besitzt keine direkte Schnittstelle zu SAP EPD. Ein manueller Prozess zum Erzeugen der Smart Assets für die verschiedenen Zielplatformen muss durchlaufen werden. Dadurch ist das Nutzen von Smart Assets für die Viewer Apps nicht gut geeignet. Es wird das Bindeglied zwischen XR Cloud und SAP EPD gebraucht - eine automatische Asset Pipeline, die den manuellen Prozess zum Erzeugen der Smart Assets ersetzt.

3.1.1. Performanz

Es wurde immer wieder in praktischen Tätigkeiten beobachtet, dass Smart Assets eine bessere Ladezeit haben im Vergleich zum Laden von Objekten als Runtime Asset. Das Unity Integration Toolkit entwickelt sich durchgehend weiter und existierende Benchmarks verlieren so ihre Gültigkeit. Eine Asset Pipeline kann dabei unterstützen, schneller Benchmarks zu erstellen und die tatsächlichen Verbesserungen zu messen.

3.1.2. Entwicklungsaufwand

Jeder Kunde, der für die eigenen 3D Anwendungen das Smart Asset Konzept nutzen muss, benötigt Unity Entwickler, die diese erstellen. Diese Unity Entwickler müssen Experte in Unity Entwicklung sein, und sich mit verschiedenen Zielplattformen und Render Pipelines auskennen. Eine Asset Pipeline hat das Potenzial diesen Entwicklungsaufwand zum Erzeugen von Smart Assets zu ersparen.

3.1.3. Darstellung von 3D Modellen auf mobilen Geräten

Die Hololens und Oculus Quest sind mobile Geräte. Die Oculus Quest ist beispielsweise ein Virtual Reality Gerät, welches Android als zugrundeliegende Platform nutzt. Es wird kein externer Rechner benötigt, es handelt sich um ein Gerät, welches seine eigene Rechenleistung mitbringt und kabellos betrieben werden kann. Die Oculus Quest ist das erste VR Gerät, das eine Massenadoption erlebt hat [27]. Anwendungen, die von der Oculus Quest konsumiert werden sollen, müssen ihren Ressourcenverbrauch gering halten. Bei den Modellen in SAP EPD handelt es sich oft um CAD Modelle, die teilweise zu komplex sind, um sie auf mobilen Geräten anzuzeigen. Die Asset Pipeline kann Smart Assets aus EPD Modellen so bauen, dass Optimierungen und Verbesserungen angewandt werden können. Dies ist ein einfacher Konfigurationsprozess. So brauchen Kunden keine Experten beauftragen, um spezielle leichtgewichtigen Varianten der Modelle neu zu erstellen.

3.1.4. Adaption

Kunden, die das Smart Asset Konzept nutzen und selber Smart Assets bauen möchten, sind gezwungen für jeden ihrer Entwickler eine Unity Lizenz zu kaufen. Eine Unity Lizenz kostet zum aktuellen Zeitpunkt 1800$ pro Jahr für einen Entwickler [28]. Dies kann sich als Hindernis für die Adaption der Smart Assets. Die Asset Pipeline kann auf Kundenseite Lizenzkosten sparen, Pre-Sales Prozesse verbessern und somit den Eintritt in Unity und XR erleichtern.

3.2. Schnittstellen

3.2.1. XR Cloud

Die XR Cloud wird in SAP BTP betrieben und ist bereits auf einer verteilten Infrastruktur. Daher eignet sie sich gut, um bestehende Geschäftsanwendungen zu erweitern. Die XR Cloud nimmt mehrere Aufgaben an: Zum Einen existiert die Smart Asset App, in der existierende Smart Assets eingesehen und verwaltet werden können. Zum Anderen gibt es die Scenes App, in der diese Smart Assets genutzt werden sollen. Die XR Cloud richtet sich an folgende Zielgruppen [29].

- **XR Developers**: Benutzen das Unity Integration Toolkit um Smart Assets zu erzeugen. Es können Inhalte aus anderen SAP Systemen beziehen (z.B. mithilfe des SAP EPD Unity Plugins). XR Developer machen ein Smart Asset durch Hinzufügen von C# Scripting "Smart". Sie definieren Parameter in der XR Cloud, mit dem die dynamischen Aspekte eines Smart Assets definiert werden können. Wie diese Parameter ausgefüllt werden, bestimmt der XR Creator.
- **XR Creators**: Entwickeln Ende-zu-Ende eine XR Anwendung. Sie verändern jedoch nicht die grundlegende Funktionalität von Smart Assets. Sie entscheiden, welche Smart Assets genutzt werden und wie die Parameter ausgefüllt werden. Sind die Parameter statisch, können diese direkt in der XR Cloud Weboberfläche definiert werden. Sind die Parameter dynamisch (z.B. aus anderen SAP Systemen), so bietet die XR Cloud eine Anbindung an Tricentis Tosca Commander, mit deren Hilfe das dynamische Setzen der Parameter durchgeführt werden kann (Testautomation).

In der XR Cloud existieren mehrere technische Rollen, die Nutzer einnehmen können. Es existieren folgende technischen Rollen [29].

- **Read**: Hat Lesereichte über die Scenes App und kann Smart Assets herunterladen.
- **User**: Verwaltet die XR Szenen und kann einen Szenenlauf starten.
- **Creator**: Verwaltet Smart Assets.
- **Admin**: Haben die selben Rechte wie User und Creator. Zusätzlich können Admins die XR Cloud konfigurieren (z.B. die Toska Integration).

Zur weiteren Analyse werden drei dieser Rollen betrachtet.

Rolle: User

Der User kann Smart Assets in Szenen konsumieren. In Abbildung 2.1 sitzt dieser Nutzer auf "Consumer"-Seite. Er hat ausschließlich Zugriff auf die Scenes App und dem Smart Asset Test Room. Der User kann bestimmen, welche Smart Assets in einer bestimmten Szene genutzt werden sollen. Meistens existiert noch kein passendes Smart Asset Binary für eine bestimmte Kombination aus Unity Version, Render Pipeline und Zielplattform. Aus diesem Grund muss erst ein Smart Asset in dieser Kombination vom Smart Asset Creator gebaut werden. Dies ist eine weitere Benutzerrolle in der XR Cloud (siehe 3.2.1). Der Smart Asset User meldet den Bedarf an den Creator. Der User kann das Smart Asset auch ohne existierenden Smart Asset Binary in eine Szene hinzufügen. Da es sich nur um eine Referenz handelt. Sobald das Smart Asset vom Smart Asset Creator erzeugt und hochgeladen wird, wird die Binary automatisch von der referenzierenden Anwendung genutzt.

Dies ist eine Schnittstelle, an der die Asset Pipeline ansetzen soll. Statt den Bedarf an einen Creator zu melden soll der User über ein Klick auf ein Button die Asset Pipeline starten können, der für den Benutzer ein Smart Asset in der richtigen Kombination baut. Es werden hierfür die Informationen aus der Szene genutzt. Dadurch benötigt der User keine weitere Konfiguration und kein Expertenwissen.

Rolle: Creator

Erzeugt die Smart Assets, die später in der Scenes App vom User konsumiert werden können. In Abbildung 2.1 sitzt dieser Benutzer auf "Producer"-Seite. Der Smart Asset Creator lädt bei den Smart Assets das Package hoch. Dies ist das Smart Asset Package. In diesem Package sind alle Dateien enthalten, um jederzeit Smart Asset Binaries daraus bauen zu können. Der Inhalt dieser Packages kann beispielsweise aus Modellen aus SAP EPD stammen, die vorher vom SAP EPD Plugin erzeugt wurden.

Mithilfe dieses Packages können auch andere Benutzer mit der Creator Rolle das Smart Asset bauen. Es handelt sich somit um eine Art "Quellcode" für Smart Assets. Bislang meldet der Smart Asset User informell Bedarf bei einem Creator an, der aus dem Package manuell ein Smart Asset Binary baut. Es wird der Bundler des Smart Asset Plugins genutzt. Je nach Unity Version, Render Pipeline und Zielplattform

nutzt der Creator seine Erfahrung, um die das Unity Projekt, in dem das Smart Asset gebaut wird, richtig zu konfigurieren.

Im Rahmen dieser Arbeit soll ein Teil der Aufgaben des Creators automatisiert werden. Das Expertenwissen des Creators soll in der Asset Pipeline einfließen. Es soll unter bestimmten Annahmen ein Smart Asset automatisiert gebaut und in die XR Cloud hochgeladen werden. Der Smart Asset Creator ist damit einzig dafür zuständig, das Smart Asset richtig zu konfigurieren, sodass dies von der Asset Pipeline genau so gebaut wird, wie erwartet. Es ist daher naheliegend, dass alle Parameter, die das Aussehen eines Assets beeinflussen in die Smart Asset Version einfließen, wo auch der Smart Asset Creator die Berechtigung hat. Auch soll nicht mehr ein Package als Zwischenschritt erzeugt werden. 3D Modelle sollen direkt aus SAP EPD genutzt werden. Hierfür müssen auch die EPD Parameter in der XR Cloud einfließen. Auch hier ist es naheliegend, die Smart Asset Version zu nutzen. Dadurch können auch verschiedene Konfigurationen an dem selben Modell angewandt werden, um unterschiedliche Anwendungen, Anwendungsfälle und Prozesse anzusprechen.

Rolle: Admin

Der Admin konfiguriert die XR Cloud und bestimmt sicherheitstechnische Aspekte. Der Admin kann beispielsweise festlegen, wie die Schnittstellen zu einzelnen externen Systemen konfiguriert werden soll. Im Falle der Asset Pipeline kann hier beispielsweise die Konfiguration der SAP EPD Zugangsdaten liegen.

3.2.2. SAP Enterprise Product Development

Die Modelle, aus denen Smart Assets gebaut werden sollen, kommen aus SAP EPD. Aus diesem Grund wird eine Schnittstelle zu diesem Produkt benötigt. Das XR Venture hat im Unity Integration Toolkit die Schnittstellen zu EPD als Plugin umgesetzt und stellt diese Kunden zur Verfügung. SAP EPD besitzt die Funktionalität gängige andere 3D Formate (wie FBX, OBJ) zu konvertieren [30].

3.2.3. Unity Integration Toolkit

Das Unity Integration Toolkit (UIT) besitzt Programmierschnittstellen und Tools, um das Nutzen verschiedener Cloud Dienste zu erleichtern.

SAP EPD Unity Plugin

Das SAP EPD Unity Plugin kann dafür genutzt werden, um die Modelle in ihrer richtigen Konfiguration zu laden. Dieses Plugin besitzt zustäzlich aber weitere Programmierschnittstellen, die für die Asset Pipeline in Frage kommen können.

Das Plugin besitzt Funktionalität, die speziell für Unity Umgebungen entwickelt wurden, und die bei Bedarf aktiviert und deaktiviert werden können. Eine dieser Funktionalitäten ist das Erzeugen von Prefabs. EPD Modelle, die zur Unity Laufzeit geladen werden, können mithilfe des eingebauten Prefab-Creators persistiert werden. Dadurch stehen die Modelle für weitere Verarbeitung zur Verfügung. Der XR Cloud Benutzer mit der Rolle "Creator" nutzt auch diesen Prefab Creator, um Smart Asset Packages oder direkt Smart Asset Binaries zu erzeugen.

Smart Asset Unity Plugin

Das Smart Asset Unity Plugin besitzt die Funktionalität, Smart Assets zu bauen. Es wird eine grafische Oberfläche genutzt, um den Build-Prozess zu konfigurieren und das Smart Asset zu bauen. Bislang wurde das Bauen von Smart Assets ausschließlich manuell durchgeführt. Es existiert offiziell keine Programmierschnittstelle zum automatisierten Bauen von Smart Assets und dieser Ansatz wurde in keiner bisherigen Arbeit evaluiert.

In dieser Arbeit wurde das Smart Asset Plugin analysiert um es ggf. so zu erweitern, dass das automatisierte Bauen möglich ist. Es wurde herausgefunden, dass es Klassen gibt, die zum Bauen genutzt werden können. Diese besitzen jedoch Abhängigkeiten mit UI Elementen. Diese Abhängigkeiten machen es nicht möglich, Smart Assets automatisiert zu bauen, da die Unity UI zum Bauen und Anzeigen von Fehlern/Ergebnissen benötigt wird. Da wir in der Pipeline aus der Kommandozeile bauen, müssen die Abhängigkeiten zu diesen entfernt werden. Aus diesem Grund muss das Smart Asset Plugin modifiziert werden, sodass das Bauen aus Programmierschnittstellen ermöglicht wird.

3.2.4. Azure Devops

Das XR Venture nutzt Azure Devops um existierende Unity Projekte vom Unity Integration Toolkit automatisiert als Plugin zu exportieren. So wird nicht der Quellcode vom UIT an Kunden geliefert, sondern das Plugin. Dieser Prozess wird bei jedem Git-Commit durchlaufen, sodass immer eine aktuellste Version des Plugins vorhanden ist. Azure Devops kann vielfältig genutzt werden. Es wird eine azure-pipeline.yml-Datei zum zentralen Konfigurieren einer Pipeline genutzt. Software Artefakte, die erzeugt werden, können in ein Artefact Storage veröffentlicht werden. Im Beispiel des UITs werden die Plugins von diesem Artefact Storage manuell heruntergeladen und zur Verteilung an jeweilige weitere Schnittstellen oder zur Analyse weiter verwendet.

Im XR Venture werden bislang von Azure Devops bereitgestellte virtuelle Maschinen genutzt. Diese Maschinen sind bei jedem Prozess frisch aufgesetzt und sorgen dafür, dass ein Build-Prozess lange dauert, da jedes Mal alle nötigen Tools neu installiert werden müssen. Wir möchten mit der Asset Pipelien eine große Menge an Unity Versionen ansprechen. Es ist daher zu evaluieren, wie eine eigene virtuelle Maschine in Azure Devops genutzt werden kann, in dem die nötigen Anwendungen bereits vorinstalliert sind.

Ebenfalls existiert bislang keine Schnittstelle von Azure Devops in andere Systeme. Der Prozess nach dem Bauen von Software Artefakten in Azure Devops ist bislang manuell. Es sollen die Azure Devops REST-Schnittstellen [31] zur Kommunikation mit anderen Systemen evaluiert und genutzt werden. Dies kann beispielsweise dafür genutzt werden, um den aktuellen Status eines Build-Prozesses abzufragen, um dies dem XR Creator (3.2.1) darzustellen, der das jeweilige Smart Asset angefragt hat. Nach erfolgreichem Abschluss eines Prozesses soll das Software Artefakt automatisch in die XR Cloud hochgeladen werden, sodass dieses Smart Asset vom XR Creator (3.2.1) genutzt werden kann.

3.3. Experteninterviews

Es wurden Experteninterviews durchgeführt, um ein tiefergehendes Verständnis über eine Pipeline zu gewinnen. Dabei wurden drei Experten im Unity Umfeld bei SAP mit Fragen rund um Unity, Pipeline und der Systemlandschaft konfrontiert. Die Fragen konzentrieren sich insbesondere auf die Artefakte und Parameter, die von einer Pipeline konsumiert werden, den Artefakten, die von der Pipeline produziert werden, und wie die Systemlandschaft aussieht. Die Interviews unterstützen dabei auch, die Limitierungen einer Asset Pipeline zu offenbaren und wo man Annahmen treffen muss.

3.3.1. Automatisierung

Alle Experten haben übereinstimmend die Vorstellung geteilt, dass eine Asset Pipeline den Prozess zum Erzeugen bestimmter Assets automatisiert (A.0.1, A.0.2, A.0.3). Die Pipeline kann automatisierte und nicht-automatisierte Elemente beinhalten, weil sich nicht alles automatisieren lässt (A.0.2, A.0.3). Es existiert eine Warteschlange, in der die Aufgaben abgearbeitet werden. Sie soll die Arbeitszeit des Mitarbeiters reduzieren (A.0.2).

3.3.2. Ausgangszustand

Man muss Modelle vorher manuell importieren und bauen (A.0.1, A.0.2). Es besteht manuelle Nacharbeit von Modellen, es gibt keinen einheitlichen Standart, der international sagt, wie groß ein Objekt ist (A.0.3). Beim Bauen von Assets wird der Entwicklerrechner blockiert (A.0.2).

3.3.3. Eingangsdaten

Die Eingangsparameter sind abhängig davon, was für eine Pipeline man haben möchte. Alle Experten haben sehr unterschiedliche Antworten mitgeteilt.

Mögliche Parameter: Unity Version (A.0.1), Zielplattform (A.0.1), Rendering Pipeline (A.0.1), Skalierung (A.0.1), Position (A.0.1), Rotation (A.0.1). Shader (A.0.2), Priorität (A.0.2).

Modelle können aus einem anderen System kommen, dann sind die Parameter für dieses System nötig A.0.1. Modelle können aus SAP EPD kommen (A.0.2).

Eine Asset Pipeline sollte in der Lage sein, die gängigen Formate, wie FBX, OBJ zu beherrschen (ggf. per Preprocessing) (A.0.3).

3.3.4. Ausgangsdaten

Auf die Frage, welche Ausgabe die Pipeline haben kann, wurden verschiedene Antworten angegeben.

- Asset Bundle / Smart Asset Binary (A.0.1, A.0.2)
- Unity Package / Smart Asset Package (A.0.1, A.0.2)
- Unity Prefabs (A.0.3)
- Andere Formate (z.B. FBX, OBJ) (A.0.2, A.0.3)

3.3.5. Render Pipelines

Es sollen die Built-In Render Pipeline und URP unterstützt werden (A.0.1, A.0.2). HDRP ist optional (A.0.1, A.0.2). Die Unterstützung von URP und HDRP geht aber nur zu einem bestimmten Grad, die Shader dieser Pipelines können nicht einfach umgewandelt werden (A.0.2). Man kann bei diesen aber eine bestimmte Grundfunktionalität unterstützen (A.0.2).

Die Scriptable Render Pipeline sollte nicht unterstützt werden, da Nutzer dieser Pipeline nicht die Zielgruppe einer Asset Pipeline sind (A.0.1).

3.3.6. Zielplattformen

Alle gängigen Plattformen sollten unterstützt werden: Windows, Mac, Android, iOS, UWP. Alle weiteren Plattformen sind eher optional (A.0.1, A.0.2). Die vollständige Unity Palette außer Exoten wie TVOS oder Nintendo (A.0.3).

3.3.7. Unity Versionen

Eine Asset Pipeline soll folgende Unity Versionen unterstützen.

- 2019 und 2020 (A.0.1)
- der LTS Stream von 2019 und 2020, optional den neuesten 2021 Stream (A.0.2)
- die neuesten Versionen von 2019 und 2020, auch die neueste Beta (A.0.3)

3.4. Aufgabenstellung

Die Ergebnisse der Interviews haben wertvolle Informationen darüber gegeben, was alles möglich ist. Alle möglichen Eigenschaften einer Asset Pipeline abzudecken würde den Rahmen dieser Arbeit sprengen und müssen daher eingeschränkt werden. In Austausch mit dem Product Owner wurde die User Story und Anforderungen festgelegt.

3.4.1. User Story

SAP EPD Kunden wollen ihre CAD Modelle auf verschiedenen Zielplattformen und Technologien, wie VR Headsets, Hololens usw. darstellen. Zum aktuellen Zeitpunkt unterstützt SAP EPD nur die Darstellung im Browser. Eine 3D Asset Pipeline für SAP EPD stellt sicher, dass diese Modelle in alle von Unity unterstützten Plattformen automatisiert umgewandelt werden können.

3.4.2. Funktionale Anforderungen

Versionsverwaltung

Änderungen an Modellen sollen in unterschiedlichen Versionen der Assets resultieren. Über eine Versionsverwaltung können Modifikationen an Modellen durchgeführt und verwaltet werden. Smart Assets können bereits versioniert werden. Die Asset Pipeline soll idealerweise an dieser Versionisierung anknüpfen und diesen erweitern.

Steuerung

Der Benutzer muss die Möglichkeit, die Asset Pipeline über eine graphische Oberfläche zu steuern. Das Starten, Stoppen und einsehen des Status der Pipeline muss ersichtlich sein. Bislang wird für alle Azure Devops Aktivitäten die Weboberfläche von Azure Devops genutzt. Es ist nicht möglich, Kunden Zugang zur Azure Devops Weboberfläche zu geben. Es muss daher ggf. eine eigene Weboberfläche erstellt werden, in der die Pipeline ohne weitere zusätzliche Authentifizierung durchgeführt werden kann. Idealerweise direkt in der XR Cloud.

Zustand der Pipeline

Der Benutzer soll den Zustand der Pipeline einsehen können. Ähnlich wie bei der Steuerung wird eine eigene Weboberfläche benötigt, in der der Zustand der Pipeline ersichtlich wird.

Zielplattformen

Folgende Zielplattformen sollen unterstützt werden.

- Android
- iOS
- MacOS
- Windows
- Universal Windows Platform
- WebGL

Für diese Zielplattformen baut das XR Venture zum aktuellen Zeitpunkt Apps. Alle weiteren Plattformen sind nicht im Leistungsumfang der Asset Pipeline.

Rendering

Die Rendering Pipeline soll vom Benutzer beeinflusst und bestimmt werden können. Der Umfang der Umsetzbarkeit dieser Anforderung ist wegen (3.3.5) offen.

Es kann aber für die folgenden drei Render Pipelines eine Basisfunktionalität zum Umwandeln und Nutzen festgelegt werden.

- Built-In Render Pipeline
- Universal Render Pipeline
- High Definition Render Pipeline

3.4.3. Nichtfunktionale Anforderungen

Onboarding

SAP Kunden brauchen keine Unity Lizenz, um Smart Assets zu bauen und nutzen. Kunden erleben dadurch ein besseres Onboarding. Kunden können ihre SAP EPD Modelle idealerweise direkt ohne Unity Entwicklung in einer XR Viewer Anwendung darstellen (z.B. dem Smart Asset Test Room oder dem MR Viewer).

Verteiltheit

SAP bedient weltweit Kunden. SAP Produkte haben nach ihrem Release eine breite Kundenbasis. Die angeforderte Lösung soll in einer skalierbaren verteilten Infrastruktur laufen, um auch die weltweite Veröffentlichung von XR Produkten mit Smart Asset Integration zu unterstützen.

Integration

Das XR Venture nutzt SAP EPD, die XR Cloud und Azure Devops in ihrer Systemlandschaft. Die Asset Pipeline soll sich in die vorhandene Systemlandschaft integrieren. Alle Asset Pipeline Aufgaben sollen die XR Cloud durchführbar sein.

Kapitel 4

Implementierung

4.1. Übersicht

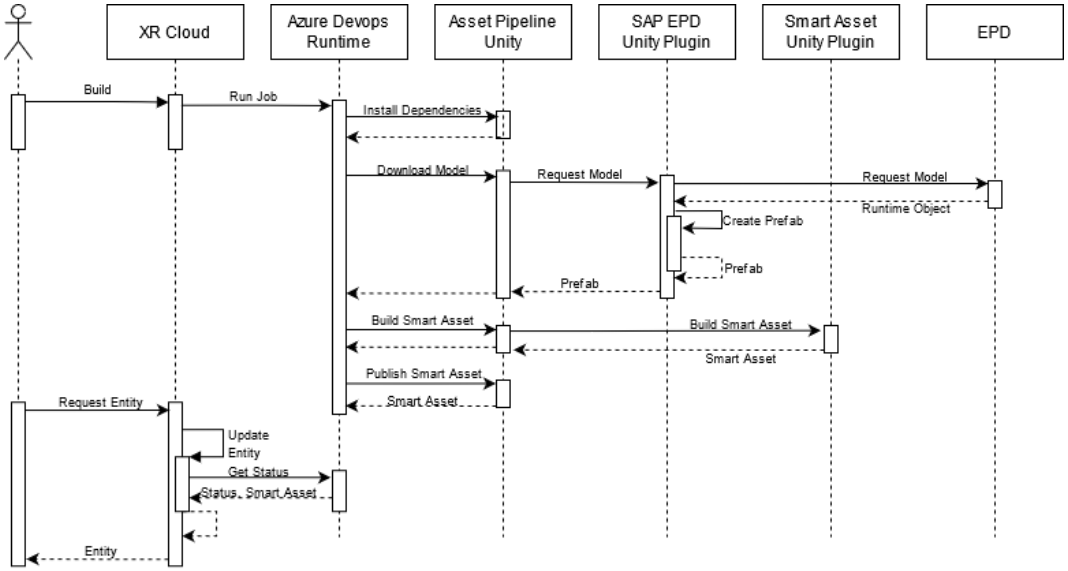


Abbildung 4.1.: Übersicht der Akteure, die bei der Pipeline zusammen spielen

4.2. Implementierungsstatus

Die Asset Pipeline besteht aus mehreren Teilen und integriert sich an mehrere Punkte in der Systemlandschaft. In 4.1 ist ein Sequenzdiagramm mit allen Akteuren.

4.2.1. Asset Pipeline Unity

Bei der Asset Pipeline handelt es sich um ein Unity Projekt, welches in einer virtuellen Machine (VM) betrieben wird. Die Pipeline besitzt Kommandozeilenschnittstellen zum Ausführen bestimmter Aktionen. Diese Schnittstellen können mit Parametern angestoßen werden. Die Pipeline durchläuft auf Unity seitig drei Phasen. Das Asset Pipeline Unity Projekt ist modular aufgebaut und besitzt einzig eine Schnittstelle zu SAP EPD, von der die Modelle bezogen werden.

Phase 1: Install Unity Packages

Im ersten Schritt findet die Initialisierung des Unity Projektes statt. Bei jedem Aufruf eines Pipeline Builds wird das Projekt in der VM neu aufgebaut. Das heißt, dass das Projekt aus dem internen SAP Git-Verzeichnis in die VM geladen wird. Das Projekt ist dadurch bei jedem neuen Aufruf der Pipeline im selben Anfangszustand. Zum Nutzen verschiedener Render Pipelines werden bestimmte Packages benötigt. Packages sind hierbei Pakete, die von Unity bereitgestellt werden und die Unity Entwicklungsumgebung um neue Eigenschaften erweitern. Im Normalfall findet die Installation dieser Pakete über den Package Manager statt, welches eine grafische Oberfläche ist. Im Falle der Asset Pipeline wurde die Funktionalität entwickelt, Packages über die Kommandozeileschnittstelle zu installieren. Hierbei wurden die Programmierschnittstellen des Package Managers genutzt. Es ist auch möglich Packages im Vornherein zu installieren und sie im Git-Verzeichnis mit zu liefern. Dieser Ansatz funktioniert in diesem Fall jedoch nicht. Die genutzte Unity Version ist dynamisch. Es wird für jede unterschiedliche Unity Version andere Package Versionen genutzt und Kombinationen miteinander können inkompatibel sein. Damit eine funktionierende Kombination zwischen Unity Version und Package Version gewährleistet werden kann, muss die Installation der Packages dynamisch erfolgen. Der Package Manager wählt hierbei selbstständig die richtige Version des Packages. Die im Rahmen dieser Arbeit entwickelte Installationsroutine für Packages kann beliebig mit weiteren Packages erweitert werden.

Phase 2: Download des EPD Modells

In der zweiten Phase findet das Herunterladen und Persistieren des Modells aus der SAP EPD statt. Dafür müssen zum einen verschiedene Parameter persistiert und angewandt werden. Über die Render Pipeline Parameter wird die Routine zum Anwenden der richtigen Render Pipeline aufgerufen. Die Render Pipeline hat großen Einfluss auf das ganze Projekt. Näheres dazu ist in 4.2.1 erläutert. Mit dem Build Target wird bestimmt, für welche Platform das Model später gebaut werden soll.

Mit den EPD Parametern werden die Parameter für das EPD Plugin bestimmt. Ist die Konfiguration angewandt, wird Unity in den Playmode gewechselt und mithilfe einer weiteren Routine automatisch mit der jeweiligen Konfiguration geladen. Ist das Laden des Modells abgeschlossen, wird die Create-Prefab Routine des EPD Plugins aufgerufen und das Modell in der Szene wird als Prefab persistiert. Zur Umsetzung dieser Phase waren Modifikationen am EPD Plugin nötig. Diese werden in 4.2.4 näher erläutert.

Phase 3: Build Smart Asset

Nach Abschluss der dritten Phase liegt das angefragte Modell als Prefab vor. Dies kann nun in der letzten Phase mithilfe des Smart Asset Plugins zu einem Smart Asset Binary gebaut werden. Zur Bestimmung der Zielplattform wird die zuvor gespeicherte Konfiguration genutzt. Zur Benutzung des Smart Asset Plugins in der Asset Pipeline waren geringfügige Änderungen an diesem nötig. Diese werden in 4.2.5 erläutert.

Render Pipeline

Damit ein Smart Asset Binary in einer Unity Anwendung genutzt werden kann, muss auch die Render Pipeline betrachtet werden.

In den Projekteigenschaften wird die Render Pipeline eines Unity Projektes festgelegt. In der Asset Pipeline wurden hierfür Pipeline Assets für URP und HDRP mitgeliefert, die je nach Render Pipeline genutzt werden. Ist die Render Pipeline festgelegt, interpretiert Unity die Shader an den Materialien mit der ausgewählten Pipeline. Nach einem Wechsel der Render Pipeline sind Objekte meistens in Pink dargestellt, weil die falschen Shader eingestellt sind. Unity bietet für die Umwand-

lung der richtigen Shader ein Werkzeug an. Mit diesem Werkzeug werden die Shader umgewandelt, sodass sie wieder in ihren richtigen Materialien dargestellt werden. Das automatisierte Umwandeln von Shadern ist problematisch: Shader haben unterschiedliche Parameter, die sich unterschiedlich auf das Material auswirken. Das Umwandeln einfacher Shader ist möglich, sollen aber komplexere Shader mit verschiedenen Parametern genutzt werden, kann sich dieser Prozess problematisch erweisen.

In der Implementierung der Asset Pipeline wurden nicht die Umwandlungswerkzeuge von Unity genutzt. Es wurde ein eigener Umwandler entwickelt. Für die EPD Modelle kommen zwei verschiedene Materialien in Frage, die einfache Shader haben. Der entwickelte Umwandler bietet Basis für weitere spätere Erweiterungen und ermöglicht genaue Aussagen darüber zu treffen, welche Shader tatsächlich unterstützt werden und welche nicht.

Für HDRP wird eine spezielle Konfiguration der Szene gebraucht, die in der Pipeline so abgedeckt ist, dass es eine spezielle Unity Szene für diese Pipeline gibt. Beim Wechsel der Render Pipeline auf HDRP wird auch ein Szenenwechsel durchgeführt.

4.2.2. Azure Devops

Für das Orchestrieren des Asset Pipeline Unity Projektes und der Schnittstelle mit der XR Cloud wird Azure Devops genutzt. Azure Devops ist vergleichbar mit anderen Development Operations Produkten wie Jenkins. Es handelt sich um ein Cloud Dienst, welches von Microsoft Azure bereitgestellt wird. Das XR Venture hat Azure Devops bereits zuvor für für Automatisierungsaufgaben genutzt. So wird beispielsweise das Smart Asset Plugin und SAP EPD Unity Plugin ebenfalls mithilfe einer Azure Devops Pipeline gebaut. Azure Devops nutzt die Beschreibungssprache Yaml zur vollständigen Konfiguration einer Pipeline. Die Azure Devops Pipeline der Asset Pipeline durchläuft im Groben drei Phasen.

Phase 1: Installiere Unity

Jedes Smart Asset muss für eine bestimmte Unity Version gebaut werden. Diese Unity Version muss auf der Virtuellen Machine, auf der die Asset Pipeline betrieben wird, installiert werden. Zur Installation der richtigen Unity Version ist es nicht

nur ausreichend, die Versionsbezeichnung (wie z.B. 2019.4.20f1) zu kennen. Es muss auch das Changeset bekannt sein. Das Changeset ist ein Hash um den richtigen Download zu identifizieren. Unity veröffentlicht durchgehend Versionen, die teilweise die gleiche Bezeichnung haben können. Um diese zu unterscheiden, wird das Changeset genutzt. Es ist möglich über bestimmte Schnittstellen, die von Unity angeboten werden, diese anzufragen.

Es haben bereits frühere Entwickler sich mit dieser Thematik beschäftigt. Es existiert zu diesem Anlass Open-Source Projekt wie DragonBox/u3d, um das Prüfen und Installieren zu automatisieren. Während der Evaluierung dieser Software ist bekannt geworden, dass sie unter der aktuellen Windows Version zum Installieren und Anfragen existierender Unity Versionen nicht mehr funktionsfähig ist [32]. Nach der Suche nach dem Problem wurde herausgefunden, dass ein bestimmter Systemaufruf einen Fehler verursacht. Einzig die Schnittstelle zum Anfragen der Changesets funktioniert, da diese keine Systemaufrufe nutzt. Im Rahmen dieser Arbeit wurde diese Schnittstelle genutzt, um eine Liste mit den Changesets zu erhalten. DragonBox/u3d wählt basierend auf das Betriebssystems die richtigen Changesets.

Das Prüfen und Installieren von Unity Versionen wurde mithilfe von Unity Hub implementiert. Das Nutzen von Unity Hub ist der primäre weg, wie Unity auf Zielrechnern installiert werden kann [33]. Unity Hub bietet ebenfalls Kommandozeilen-Schnittstellen zum Installieren von Unity Versionen. Diese sind nicht in der Web-Dokumentation von Unity dokumentiert. Erst durch das Nutzen des "help"-Kommandos von Unity Hub wird ersichtlich, welche weiteren Schnittstellen für Unity Hub existieren. Diese Kommandizeilen-Schnittstellen eignen sich besonders gut für den Anwendungsfall, weil man so die Installation in exakt der selben Konfiguration durchführen kann, wie sie in der Unity Hub grafischen Oberfläche genutzt werden. Das ermöglicht eine nahtlose Verwaltung der Installationen, sowohl in Kommandozeile, als auch in der grafischen Oberfläche.

Im Rahmen dieser Arbeit wurde eine Reihe von Kommandozeilen geschrieben, die prüfen, ob eine Unity Version existiert, falls nicht, den richtigen Changeset zu einer Unity Version abfragen und diese inklusive der richtigen Module installiert.

Phase 2: Führe Asset Pipeline Unity aus

In der zweiten Phase werden alle Schritte aus des Asset Pipeline Unity Projektes ausgeführt, um Abhängigkeiten in Unity zu installieren, das Modell herunterzuladen und das Smart Asset zu bauen. Dies ist in 4.2.1 detailliert erläutert.

Phase 3: Veröffentliche Logs und Smart Asset

Zum Abschluss werden die veröffentlichten Unity Logs und das erzeugte Smart Asset in Azure Devops im Artefact Storage veröffentlicht. Die XR Cloud hat über REST-Schnittstellen Zugriff auf diesen Storage und kann auf die Dateien zugreifen.

Virtuelle Maschine

Für das Nutzen einer eigenen VM in Azure sind zusätzliche Konfigurationen nötig. In Azure Devops wird hierfür bei den Organisationseigenschaften ein Pool definiert. Ein Pool ist eine Menge an Virtuellen Maschinen, die zum Einsatz kommen können, wenn es Einträge die Pipleline Warteschlange gibt. Ist ein offener Pipeline Auftrag vorhanden, so wird die nächste freie Maschine im Pool genutzt. In der Implementierung dieser Arbeit wurde eine Maschine in dem jeweiligen "Default" Pool konfiguriert. Zum Skalieren kann die Menge der Maschinen erhöht werden.

Auf der VM sind folgende Anwendungen installiert.

- **Azure Devops Agent**: Zur Anbindung der Maschine an die Pipeline.
- **Ruby 3.x.x**: Zum Nutzen von DragonBox/u3d.
- **DragonBox/u3d**: Zur Abfrage der Unity Changesets.
- **Unity Hub**: Zur Installation von Unity Versionen.
- **Windows 10 SDK**: Wird für das Bauen von Smart Assets benötigt, die sich an UWP wenden.

Sind diese Anwendungen installiert, und der Azure Devops Pool konfiguriert, dann kann jeder Windows-Rechner, die vom Internet aus erreichbar ist, für die Pipeline genutzt werden.

4.2.3. XR Cloud

Asset Pipeline Service

Es wurde ein neuer XR Cloud Service für die Asset Pipeline benötigt. Dieser Service nimmt alle Anfragen in Hinsicht auf Asset Pipeline entgegen und verarbeitet diese weiter. Es gibt zwei relevante Schnittstellen.

Update Status: Die Update Status Schnittstelle sorgt dafür, dass der Zustand der Asset Pipeline durchgehend aktualisiert wird. Diese Schnittstelle wird beim Aufruf der Smart Asset Version Entitäten automatisch aufgerufen. Es ist auch möglich diese Schnittstelle zum Debugging oder Testen manuell aufzurufen.

Trigger Build: Das ist die eigentlische Schnittstelle um Pipeline Aufträge zu starten. Diese Schnittstelle nimmt vier Parameter entgegen.

- Smart Asset Version ID: Die ID der Smart Asset Version, für die ein Smart Asset gebaut werden soll.
- Build Target: Die Zielplattform, für die gebaut werden soll.
- Render Pipeline: Die ausgewählte Render Pipeline.
- Unity Version: Die ausgewählte Unity Version.

Diese Schnittstelle wird auch beim Erzeugen mehrerer Pipeline Aufträge hintereinander aufgerufen. Die Pipeline Aufträge werden in die Pipeline Warteschlange ergänzt, es wird ein Pipeline Durchlauf über die Azure REST-Schnittstellen angestoßen. Als Referenz zum Durchlauf wird die Azure Devops Build Id in der XR Cloud Datenbank hinterlegt.

Smart Asset Pipeline App

Hierbei handelt es sich um eine neue App, die im Rahmen dieser Arbeit entwickelt wurde. Diese App ist über das XR Cloud Launchpad zu erreichen (siehe 4.2).

Mithilfe der Smart Asset Pipeline kann der Status der Pipeline eingesehen werden. Es ist sichtbar, wie viele und welche Aufgaben in der Warteschleife sind. Es können Pipeline Aufträge abgebrochen werden (siehe 4.3).

Es wurde eine weitere Ansicht implementiert, in der neue Pipeline Aufträge als Batch-Job gestartet werden können. Es wurde eine Suchhilfe implementiert, da-

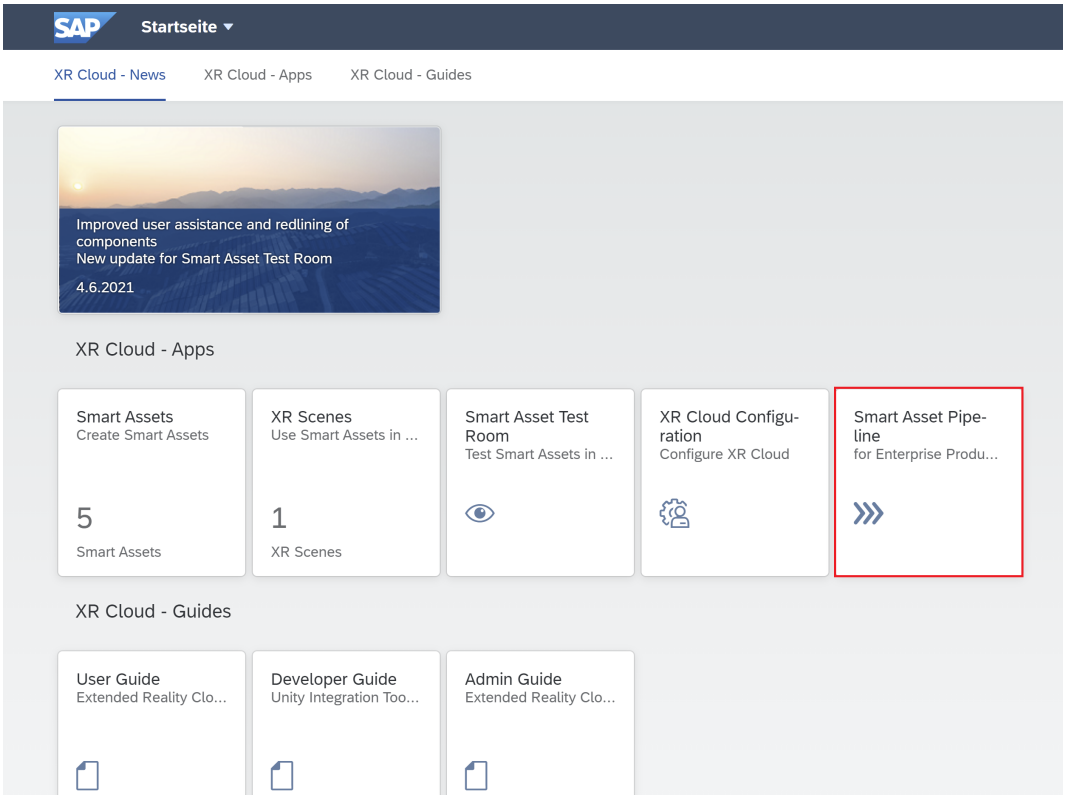


Abbildung 4.2.: Die Asset Pipeline App im XR Cloud Launchpad

mit der Nutzer einfach die gewünschte Kombination der Parameter auswählen kann (siehe 4.4).

Die Suchhilfe für Unity Versionen wurde so implementiert, dass man sowohl vordefinierte Unity Versionen nutzen kann, als auch Versionen eintragen kann, die nicht in der XR Cloud vordefiniert sind. So wird die Asset Pipeline für jede beliebige Unity Version nutzbar. Die Evaluierung über die Korrektheit der Versionsbezeichnung findet in Azure Devops bei der Unity Installationsroutine statt (siehe 4.2.2).

Konfiguration App

Die XR Cloud Konfiguration wurde mit der Asset Pipeline Konfiguration erweitert, sodass die Zugangsdaten zur Asset Pipeline Laufzeitumgebung vom Administrator eingepflegt werden können (siehe 4.5). Die eingegebenen Werte in der Asset Pipeline App werden beim Speichern mithilfe eines Verbindungstests geprüft. Die Eingaben werden abgelehnt, falls der Verbindungstest fehlschlägt (und es sich so-

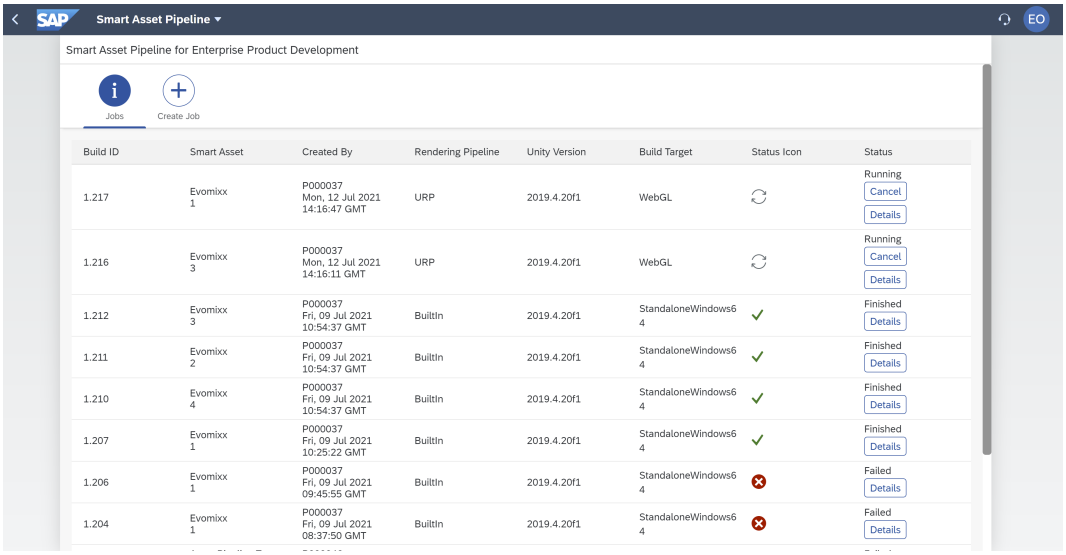


Abbildung 4.3.: Die Übersicht der Pipeline Aufträge in der Asset Pipeline App

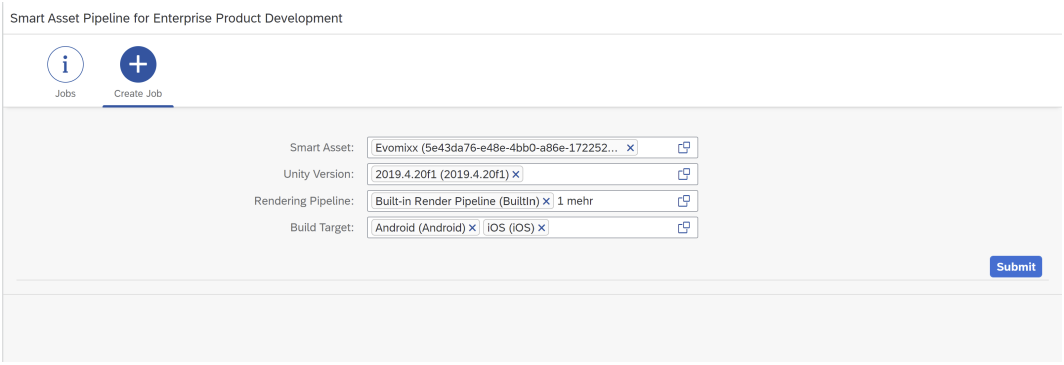


Abbildung 4.4.: Formular zum Erzeugen von Batch Jobs in der Asset Pipeline App

mit um eine fehlerhafte Eingabe handelt). Die Eingaben werden gespeichert, falls der Verbindungstest erfolgreich ist.

Ebenfalls ist sie SAP Enterprise Produkt Development Konfiguration hinzugekommen, in der die Zugangsdaten der genutzten SAP EPD Cloud Instanz definiert werden (siehe 4.6).

Die Pipeline ist erst dann funktionsfähig, wenn sowohl die Asset Pipeline Konfiguration, als auch die SAP EPD Konfiguration gesetzt sind.

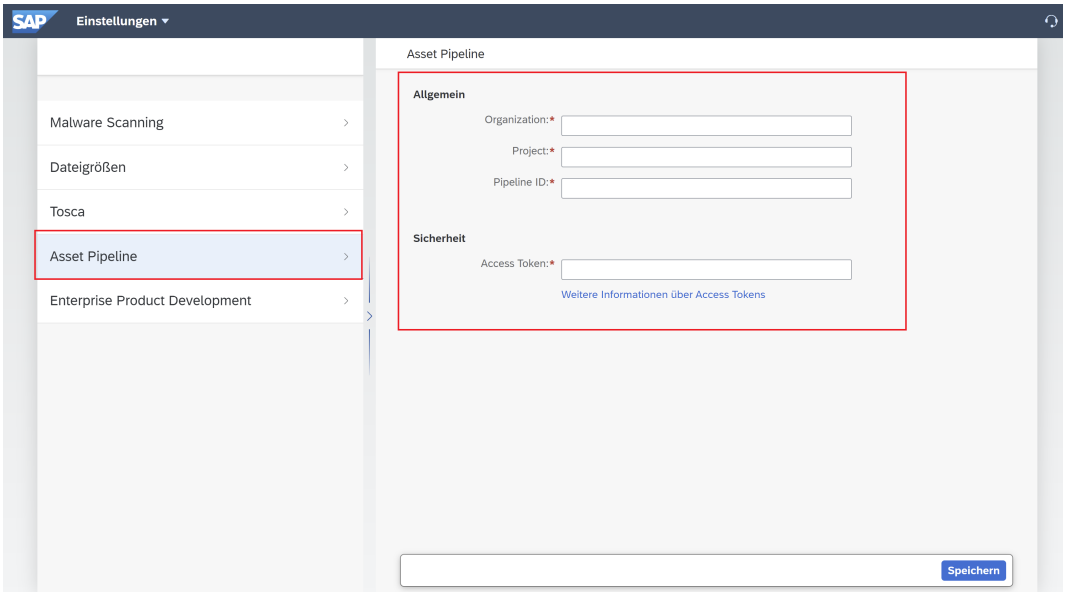


Abbildung 4.5.: Konfiguration der Asset Pipeline in der Konfigurationsapp

Smart Asset App

In der Smart Asset App wurde in den Versionen die modellspezifische Konfiguration der Asset Pipeline angelegt (siehe 4.7). Es wurden folgende Felder zum Konfigurieren der Smart Asset Versionen hinzugefügt.

- **Scene-ID (String)**: Die Scene-Id aus SAP EPD. Mithilfe dieses Wertes kann die Asset Pipeline das richtige Modell beziehen.
- **Metadaten (Boolean)**: Eine Checkbox in der angegeben werden kann, ob dem Modell die Metadaten hinzugefügt werden sollen.
- **Animationen (Boolean)**: Eine Checkbox in der angegeben werden kann, ob Animationen hinzugefügt werden sollen.
- **Skalierung (Float)**: Eine Fließkommazahl, die die Skalierung festlegt. Im Interview A.0.3 hat Experte C auf die Problematik mit uneinheitlichen Skalierungen gemacht. Mithilfe dieses Wertes kann die Skalierung über Modelle hinweg normalisiert werden.

Es können mehrere Versionen eines Smart Assets mit unterschiedlichen Konfigurationen angelegt werden. Dieses Prinzip wurde auch in 5.1 angewendet, um Benchmarks zu erstellen.

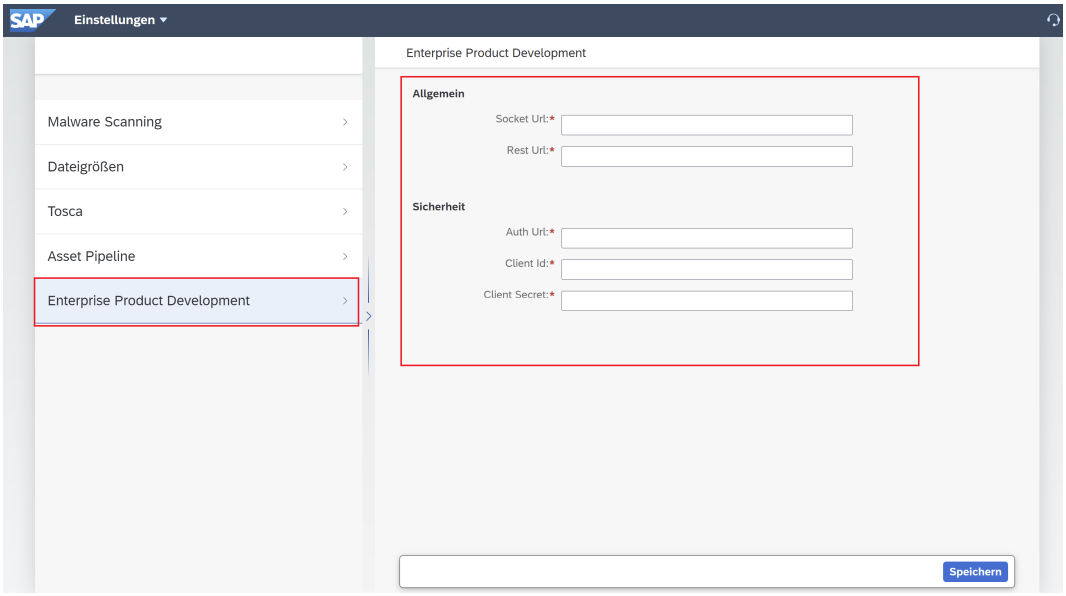


Abbildung 4.6.: Konfiguration von SAP EPD in der Konfigurationsapp

Diese Felder sind im Datenmodell einer Smart Asset Version als "struct" definiert. Es können beliebig viele weitere Felder erweitert werden. Im Ausblick 6.3 wird hierauf näher eingegangen.

Scenes App

In der Scene Manager App hat ein Nutzer die Möglichkeit, Smart Assets einer Szene zuzuordnen. Eine Szene ist dabei eine bestimmte Kombination aus Smart Assets um einen bestimmten Prozess wieder zu spiegeln. Die Suchhilfe zum Hinzufügen von Smart Assets in eine Szene wurde erweitert. In der Suchhilfe ist ersichtlich, ob es ein Build für ein Smart Asset bereits existiert. Falls nicht, kann der Benutzer das betroffene Smart Asset mit einem Mausklick bauen (siehe 4.8).

Falls ein Smart Asset bereits in Auftrag ist, ist ersichtlich, dass dieser in Arbeit ist.

4.2.4. SAP EPD Unity Plugin

Es mussten Änderungen am SAP EPD Unity Plugin durchgeführt werden. Für das Erzeugen von Unity Prefabs aus den geladenen Modellen besitzt das SAP EPD Unity Plugin ein Werkzeug, den "Prefab Creator". Dieses Werkzeug ist bislang ausschließlich für die manuelle Benutzung über den Editor konzipiert. Bislang bestand

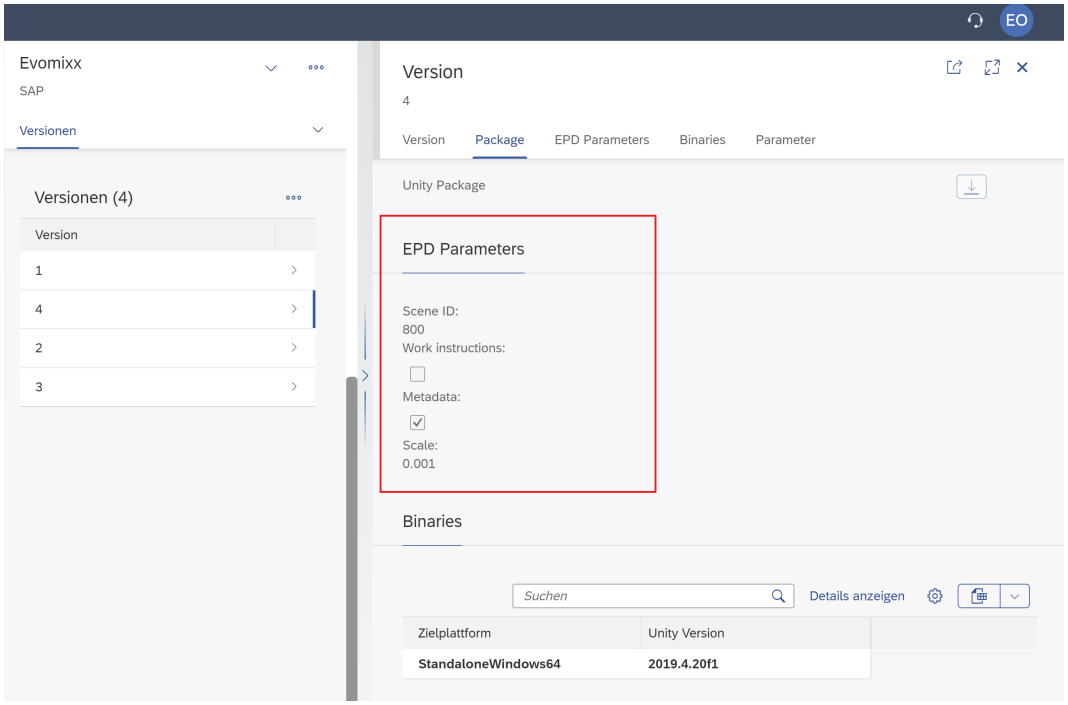


Abbildung 4.7.: Konfiguration der Smart Asset Version

kein Bedarf automatisiert Prefabs aus Modellen zu erzeugen. Dieses Werkzeug wurde mit statischen Schnittstellen erweitert, sodass die Pipeline dieses Werkzeug ebenfalls nutzen kann und keine grafischen Elemente benötigt.

4.2.5. Smart Asset Unity Plugin

Das Smart Asset Unity Plugin besitzt Entwicklerschnittstellen zum Erzeugen von Smart Assets. Man kann die Abhängigkeiten und Parameter definieren und so ein Smart Asset bauen. Diese Entwicklerschnittstelle besitzt jedoch eine Abhängigkeit auf eine UI Komponente. Meldungen, die das Plugin erzeugt, können ausschließlich über die abhängigen UI Komponenten dargestellt werden. Im Rahmen dieser Arbeit wurde diese Abhängigkeit von UI Komponenten bei der Smart Asset Erzeugung aufgelöst und ein generischer Ansatz für das Behandeln der Meldungen eingesetzt. Die Asset Pipeline hat so die Möglichkeit Meldungen selber zu behandeln und den Prozess zum Bauen von Smart Assets zu automatisieren.

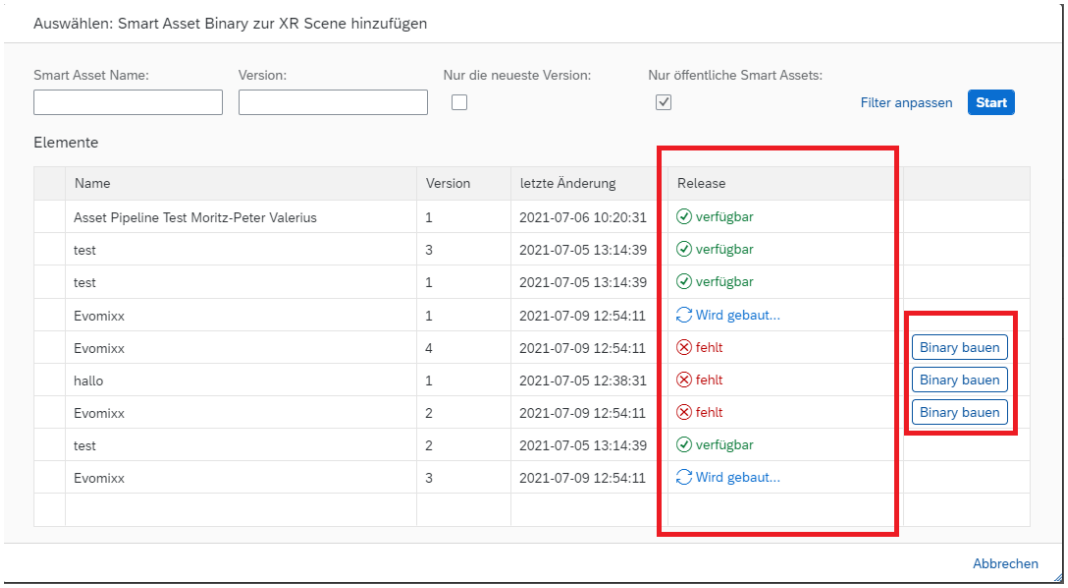


Abbildung 4.8.: Suchhilfe für das Hinzufügen von Smart Assets in eine XR Scene

4.3. Verwendete Software

- **Unity**: Unity wird als Platform für das Bauen der Smart Assets genutzt.
- **Unity Hub**: Unity Hub wird zum Installieren benötiger Unity Versionen mit deren Modulen genutzt.
- **DragonBox/u3d** [34]: Ursprünglich wurde diese Software aufgenommen, um das Installieren fehlender Unity Versionen zu automatisieren. Sie ist genau für den Anwendungsfall gedacht, dass Unity in einer Pipeline genutzt werden muss.
- **Azure Devops Agent** [35]: Für die Kommunikation von Azure Devops mit der VM ist der Azure Devops Agent nötig. Dies wird auf der VM installiert und in Azure Devops konfiguriert.

4.4. Verwendete Hardware

- **Virtuelle Machine**: Das Asset Pipeline Unity Projekt zum Bauen der Smart Assets wird in einer virtuellen Maschine betrieben. Diese VM wurde auf Microsoft Azure gebucht. Auf der VM ist Windows Server 2019 installiert.

4.5. Verwendete Cloud-Dienste

- **XR Cloud**: Ein Cloud Dienst mit Apps um
- **Business Application Studio**: Zur Entwicklung von UI5 Apps und der XR Cloud Services wurde das SAP Cloud Produkt Business Application Studio genutzt. Dies ist eine Entwicklungsumgebung die vollständig in der Cloud läuft und über den Browser zu erreichen ist.
- **SAP EPD Cloud**: Der SAP Cloud Dienst zum Speichern und Bearbeiten von 3D Modellen.
- **Azure Devops**: Ein Cloud Dienst von Microsoft zur Automatisierung von Prozessen.

4.6. Verwendete Software-Komponenten

- **Dotnet Framework**: Die Entwicklungsplatform, die bei Unity zugrunde liegt.
- **NodeJS**: Die XR Cloud ist sowohl bei den UI5 Apps, als auch bei den Services mit NodeJS entwickelt.
- **UI5**: Das Web-Framework für SAP Web-Anwendungen.
- **Smart Asset Unity Plugin**: Ein Unity Plugin um Smart Assets zu bauen oder aus der XR Cloud in Unity zu laden. Teil des Unity Integration Toolkits.
- **SAP EPD Unity Plugin**: Ein Unity Plugin um SAP EPD Modelle aus der SAP EPD Cloud inklusive Metadaten und Animationen in Unity zu laden. Teil des Unity Integration Toolkits.
- **Windows 10 SDK** [36]: Für das Bauen von Smart Assets für UWP.

Kapitel 5

Evaluierung

5.1. Performanz

Zum Ersten Mal ist es ohne größere Aufwände möglich vergleichbare Benchmarks für das Smart Asset Konzept und dem Laden von Modellen direkt aus SAP EPD zu erstellen. Der folgende Benchmark offenbart die Vorteile des Smart Asset Konzepts. Es wurde das selbe Modell mit der selben Konfiguration sowohl mit dem Smart Asset Konzept (Assets werden aus der XR Cloud heruntergeladen) als auch ohne das Smart Asset Konzept (Assets werden aus SAP EPD Cloud geladen). Für den Benchmark wurde Unity folgendermaßen konfiguriert.

- **Unity Version**: 2019.4.20f1
- **Render Pipeline**: Built-In
- **Zielplattform**: StandaloneWindows64 (Windows)

Es wurde das Evomixx-Modell verwendet (siehe 5.1). Dieses Modell besitzt im SAP EPD XR Tenant die Scene-ID 800. Es gibt hierbei sowohl Metadaten, als auch Animationen. Es eignet sich damit gut für verschiedene Versionen. Die Datenübertragungsrate im Download im gemessenen Netzwerk beträgt maximal 40MBit/s [37].

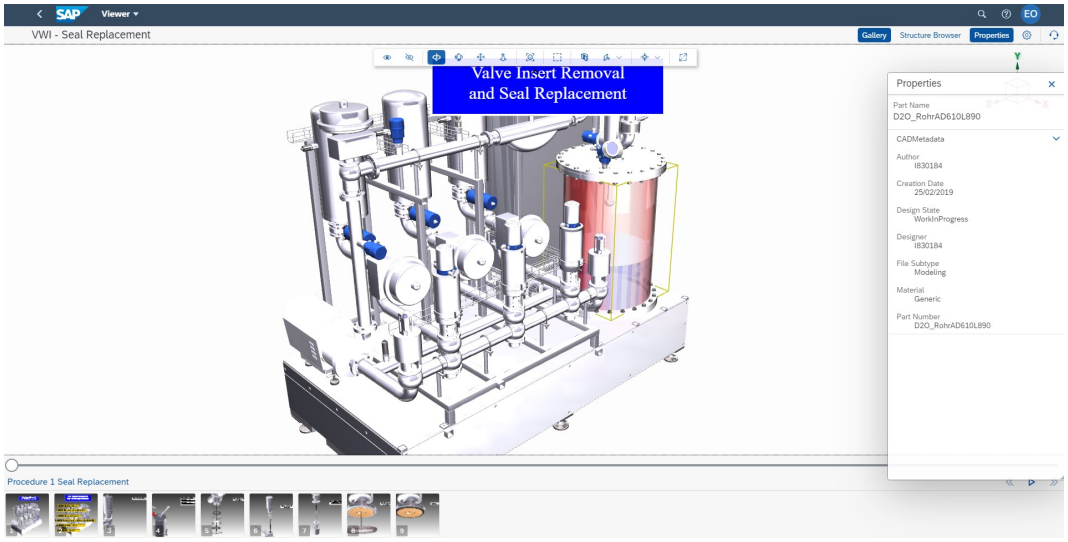


Abbildung 5.1.: Evomixx: Modell mit der Scene-ID 800 mit Metadaten und Animationen

SAP EPD / Runtime Assets

Für die Tests der Ladezeiten aus Modellen für SAP EPD wurde die aktuellste Version des SAP EPD Unity Plugins[1] genutzt. Es wurden mehrere Messungen durchgeführt. Da die Messungen stark auch von den Netzwerkbedingungen abhängen, sind die Ergebnisse als ungefähre Werte zu sehen.

Quelle	Modell	Animationen	Metadaten	Ladezeiten (Sekunden)
SAP EPD	Evomixx (Scene-ID: 800)	Ja	Ja	26,5 - 31,2
SAP EPD	Evomixx (Scene-ID: 800)	Nein	Ja	27,3 - 31,6
SAP EPD	Evomixx (Scene-ID: 800)	Ja	Nein	10,8 - 11,3
SAP EPD	Evomixx (Scene-ID: 800)	Nein	Nein	9,6 - 10,5

Tabelle 5.1.: Benchmark der Ladezeiten von Modellen als Runtime-Objekt

XR Cloud / Smart Assets

Für die Smart Assets wurde die Asset Pipeline zum Bauen verwendet. Es wurden dafür verschiedene Versionen in der XR Cloud für das selbe Smart Asset gepflegt. Es wurde auch die Asset Pipeline mit der selben SAP EPD Plugin Version aktualisiert, wie sie im ersten Teil des Tests durchgeführt wurde.

[1] Commit Hash: fd56541e899e1aea2059a825780aaddd460071ec

Quelle	Modell	Animationen	Metadaten	Ladezeiten (Sekunden)
XR Cloud	Evomixx (Smart Asset)	Ja	Ja	5,0 - 5,8
XR Cloud	Evomixx (Smart Asset)	Nein	Ja	5,1 - 5,3
XR Cloud	Evomixx (Smart Asset)	Ja	Nein	5,0 - 5,3
XR Cloud	Evomixx (Smart Asset)	Nein	Nein	4,9 - 5,4

Tabelle 5.2.: Benchmark der Ladezeiten der Modelle als Smart Asset

Beobachtung

Es lässt sich beobachten, dass Smart Assets nicht nur eine bessere, sondern auch eine konstante Ladezeit haben, unabhängig von der Funktionalität, die genutzt wird.

5.2. Entwicklungsaufwand

Kunden benötigen Unity Entwickler, die SAP EPD Modelle zu Smart Assets umwandeln. Im Kontext der XR Cloud werden diese Unity Entwickler auch XR Developers (siehe 3.2.1) bezeichnet. Die Unity Entwickler durchlaufen in diesem Prozess verschiedene Schritte. Die Asset Pipeline spart den Entwicklungsaufwand, der in diesem Prozess entsteht. Im Folgenden sind diese Schritte vereinfacht dargestellt und für jeden Prozessschritt ein ungefährer Aufwand in Personenminuten/Personenstunden festgelegt. Die berechneten Zeiten stellen den Personenaufwand für einen Unity Experten dar.

5.2.1. Unity Installation

Der XR Developer muss anfangs herausfinden, an welche Zielanwendung sich ein Smart Asset richten soll. Das Smart Asset muss exakt den Anforderungen der Zielanwendung angepasst werden. Hierbei ist zu bestimmen, welche Unity Version, welche Zielplattform und welche Render Pipeline genutzt werden soll.

Der XR Developer öffnet Unity Hub und installiert die Unity Version inklusive dem benötigten Modul. Die Dauer dieser Installation ist abhängig von der Downloadgeschwindigkeit des Netzwerkes. Um herauszufinden, in welcher Größenordnung sich solch eine Installationsdauer befindet, und ob die Installationsdauer vom Modul abhängt, wurden solche Installationen durchgeführt und die Zeit dabei ge-

messen. Die Datenübertragungsrate im Download im gemessenen Netzwerk beträgt maximal 40MBit/s [37].

Unity Version	Zielplattform	Dateigröße (MB)	Dauer (Minuten)
2020.3.13f1	Windows Build Support (IL2CPP)	374,4	15
2020.3.13f1	iOS Build Support	etwa 1600	16

Tabelle 5.3.: Dauer einer Unity Installation

Dies ist ein konstanter Aufwand, der immer nur einmal stattfindet. Potenziell muss der XR Developer im Laufe der Zeit für viele verschiedene Kombinationen durchführen. Für weitere Berechnungen wird für diesen Prozessschritt mit einem Aufwand von 15 Personenminuten (0,25 Personenstunden) bemessen.

5.2.2. Projektkonfiguration

Nachdem Unity installiert ist, müssen Abhängigkeiten installiert (z.B. Smart Asset Unity Plugin) und das Projekt konfiguriert werden. Diese Konfiguration ist stark vom Anwendungsfall abhängig. Typischerweise finden folgende Schritte statt.

- **Render Pipeline**: Für die Built-In Render Pipeline sind meistens keine weiteren Schritte nötig. Wird jedoch URP oder HDRP genutzt, so müssen als erstes die Abhängigkeiten installiert, Dateien für die Konfiguration angelegt und die Projekteigenschaften angepasst werden. Zum Schluss müssen bestehende Materialien umgewandelt werden. Im Falle vom SAP EPD Unity Plugin müssen nur zwei Materialien umgewandelt werden.
- **Smart Asset Unity Plugin**: Es muss die Konfiguration des Smart Asset Unity Plugins festgelegt werden. Diese Konfiguration bestimmt, wo das erzeugte Smart Asset später hochgeladen und veröffentlicht wird.
- **SAP EPD Unity Plugin**: Es muss eine Szene eingerichtet werden, in dem der Scene-Loader eingebunden ist. Typischerweise nutzen XR Developer hierfür das SAP EPD Unity Beispielprojekt, in dem solch eine Szene bereits fertig eingerichtet ist. In der Beispielszene wird bei der jeweiligen Komponente die Zugangsdaten für SAP EPD angelegt. Es findet außerdem die Konfiguration des Modells statt (Animationen, Metadaten).

In einer Messung wurde ein leeres Unity Projekt erstellt, in der die Zeit für die Konfigurationsschritte gemessen werden. Es wird Unity 2020.3.13f1 genutzt. Es

soll die Universal Render Pipeline für das Smart Asset genutzt werden. Es wurden die Schritte in 5.4 durchgeführt und dabei die Zeiten gemessen.

Schritt	Dauer (Minuten)
Unity Öffnen	1
Zielplattform ändern	1
Installation und Konfiguration "Universal Render Pipeline"	5
Installation und Konfiguration "Smart Asset Unity Plugin"	2
Installation und Konfiguration "SAP EPD Unity Plugin"	5

Tabelle 5.4.: Dauer von Schritte zum Erzeugen von Smart Assets, ausgeführt von einem erfahrenen XR developer

Der Aufwand zum Konfigurieren eines Unity Projektes für das Bauen von Smart Assets wird mit 14 Personenminuten (0,23 Personenstunden) bemessen.

5.2.3. Modelle herunterladen und Prefabs erstellen

Das Herunterladen von Modellen findet über das SAP EPD Unity Plugin statt, welches zuvor konfiguriert wurde. Modelle liegen anfangs als Unity Runtime Objekt vor und müssen persistiert werden, damit sie zu Smart Assets umgewandelt werden können. Zur Messung werden Modelle herangezogen, die unterschiedliche Komplexitäten besitzen. Alle Modelle werden mit Metadaten geladen.

Modell und Scene-Id	Anzahl Meshes	Anzahl Gameobjekte	Dauer Download (Minuten)	Dauer Prefaberstellung (Minuten)
Sapster (286)	4216	5003	0,5	2
Evomixx (887)	4774	7055	1	1
Kundenmodell (756)	31264	36349	1,5	15

Tabelle 5.5.: Modelle mit unterschiedlichen Komplexitäten

Kunden nutzen CAD Modelle die meistens eine deutlich höhere Komplexität besitzen, als Modelle für Showcases wie Sapster oder Evomixx. Für die weitere Berechnung wird für die das Herunterladen von Modellen und Erzeugung von Prefabs insgesamt 15 Personenminuten (0,25 Personenstunden) pro Modell herangezogen.

5.2.4. Smart Assets Erzeugen

Die erzeugten Prefabs müssen zu Smart Assets umgewandelt werden. Hierfür öffnet der XR Developer den Smart Asset Bundler, konfiguriert die Maske zum Erzeugen von Smart Assets und lässt den Prozess laufen. Nach Anschluss der Erzeugung wird das Modell mit dem Bundler in die XR Cloud hochgeladen und steht für die Nutzung von XR Creators (siehe 3.2.1) zur Verfügung. In dieser Messung werden die Smart Assets ohne Smart Asset Package gebaut.

Modell	Zielplattform	Dauer Build (Minuten)	Dateigröße Smart Asset (MByte)	Dauer Upload (Minuten)
Sapster	Windows	10	5	0
Evomixx	Windows	25	12	0
Kundenmodell	Windows	540	18	1

Tabelle 5.6.: Modelle mit unterschiedlichen Komplexitäten

Mithilfe des Smart Asset Bundlers ist der manuelle Prozess zum Bauen von Smart Assets sehr zeitintensiv. Entwickler würden typischerweise den Build-Prozess über Nacht laufen lassen oder ihn auf einem zweiten Rechner durchführen. Während des Prozesses ist der Entwicklerrechner für weitere Arbeiten mit Unity blockiert oder unperformant. Es ist schwierig für diesen Prozess den Aufwand zu bestimmen, da die Dauer dieses Prozesses stark von der Zielplattform und dem zugrunde liegenden Modell abhängt. Falls beispielsweise über ein Windows-Rechner ein Smart Asset für iOS gebaut wird, so dauert dieser Prozess sehr viel länger, als wenn das Smart Asset auf einem Windows-Rechner für Windows gebaut wird.

Der Build Prozess für Smart Assets, in dem der Entwicklerrechner blockiert ist, wird mit 4 Personenstunden berechnet, da hier insbesondere die komplexeren Modelle die meiste Zeit beanspruchen. Anders als in der oberen Messung findet das Builden des weiteren in nicht nur eine Zielplattform statt, sondern in mehrere. Auch bei weniger komplexen Modellen würde der Prozess für acht Zielplattformen 4 Stunden in Anspruch nehmen. Das Triggern des Builds und Hochladen wird mit 5 Personenminuten (0,083 Personenstunden) berechnet.

5.2.5. Vereinfachte Betrachtung

Mithilfe dieser Informationen soll eine Hochrechnung der Entwicklungsstunden für Experten-Entwickler stattfinden. Beim ersten Bauen eines Smart Assets in eine Zielplattform findet ein konstanter Aufwand zum Installieren von Unity und Konfigurieren der Entwicklungsumgebung statt. Dieser Aufwand wird für die einfache Betrachtung vernachlässigt. Es wird angenommen, dass keine Fehlerquellen vorhanden sind, gleiche Prozesse sich nicht wiederholen und der Entwicklerrechner bei den Prozessen nicht blockiert wird.

Die Kosten der Asset Pipeline werden mit 200 € pro Monat für die Virtuelle Maschine berechnet. Die Kosten für eine Personenstunde eines Unity Entwicklers wird mit 100 € berechnet. Das Erzeugen von Smart Assets muss bei jedem Durchlauf passieren. Der Aufwand von 0,083 Personenstunden für das Triggern des Builds und Hochladen des Smart Assets wird in jedem Fall übernommen. Es ergibt sich die Darstellung 5.2.

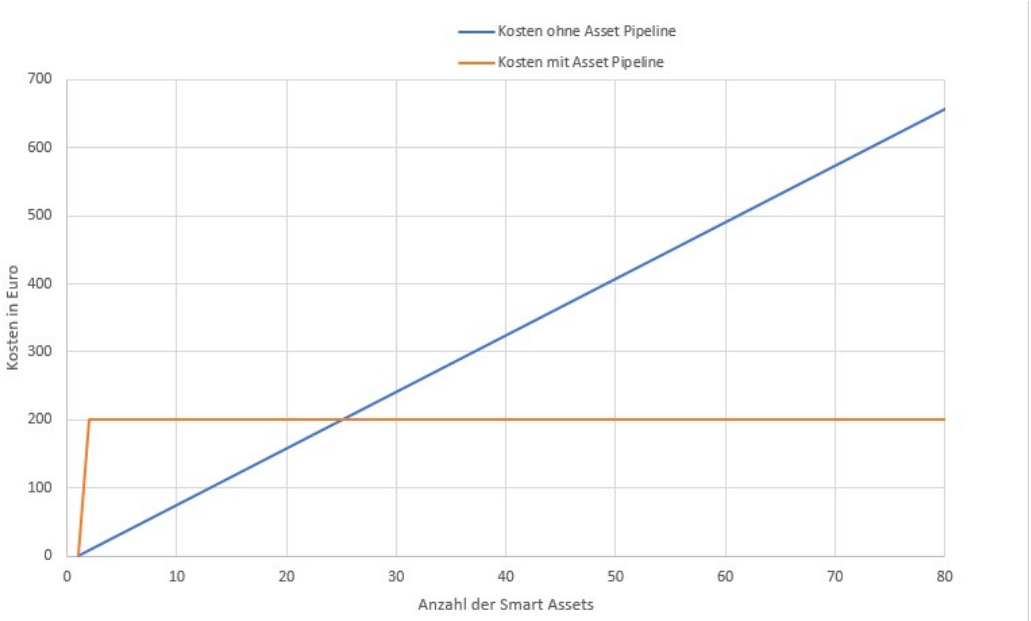


Abbildung 5.2.: Kummulierte Kosten bei einem Aufwand von 0,083 Personenstunden pro Smart Asset (B.2.1)

Der Break-Even-Point für die Asset Pipeline befindet sich bereits bei 25 Smart Assets pro Monat.

Es ist in 5.3 zu beobachten, dass der Preis pro Smart Asset für das manuelle Bauen ohne Asset Pipeline konstant bleibt, während der Preis für ein Smart Asset mit der

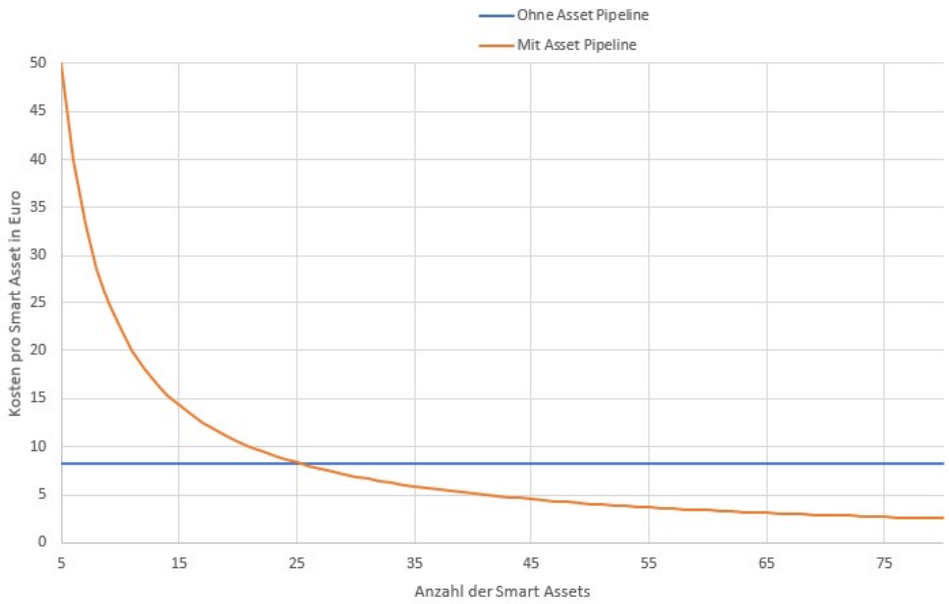


Abbildung 5.3.: Kosten pro Smart Asset bei einem Aufwand von 0,083 Personenstunden pro Smart Asset (B.2.1)

Asset Pipeline mit jedem weiteren Smart Asset geringer wird. Die Kosten mit der Asset Pipeline sind bei 100 Smart Assets bei 2 € pro Smart Asset.

5.2.6. Realistische Betrachtung

Die einfache Berechnung demonstriert eindrucksvoll, wie schnell sich die Asset Pipeline amortisiert. Diese Betrachtung ist jedoch fern von der Realität. Erfahrungen mit dem Bauen von Smart Assets zeigen, dass im manuellen Prozess viele Fehlerquellen existieren können. Oftmals wiederholen sich die zeitintensivsten Schritte mehrmals, bis ein erfolgreicher Build durchgeführt werden kann. Die kritischsten Build Prozesse drehen sich meistens um die komplexeren Kundenmodelle. In der realistischen Betrachtung wird ausgegangen, dass alle Prozessschritte bei jedem Build Prozess durchlaufen werden müssen, damit vor jedem Smart Asset Build tatsächlich ein sauberes Projekt mit jeweils den aktuellsten Smart Asset Unity Plugin und SAP EPD Unity Plugin Versionen zur Verfügung steht. Entwicklern stehen meistens keine Ersatzrechner zur Verfügung, in der die Builds parallel zur Arbeitszeit durchgeführt werden. Die Arbeitszeit wird durch den Build Prozess beeinträchtigt. Folgende Personenstunden werden für die weitere Berechnung heran gezogen.

Prozess	Personenstunden
Unity Installation	0,25
Unity Konfiguration	0,23
Modelle herunterladen und Prefabs erstellen	0,25
Smart Assets erzeugen und hochladen	4,0
Summe	**4,73**

Tabelle 5.7.: Aufwandsabschätzung für das manuelle Bauen eines Smart Assets in der realistischen Betrachtung

Dieser Wert deckt sich bei komplexen Kundenmodellen mit den Erfahrungen des XR Venture Teams. Ausgehend von einem Stundengehalt von 100 € amortisiert sich die Asset Pipeline bei der realistischen Betrachtung vom ersten Build an.

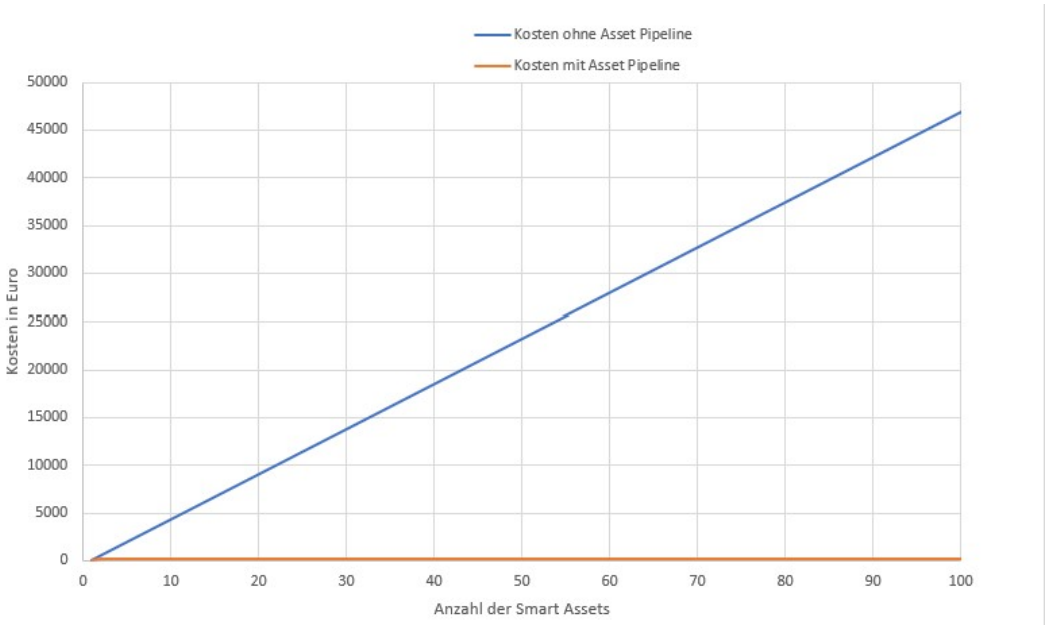


Abbildung 5.4.: Kummulierte Kosten bei einem Aufwand von 4,73 Personenstunden pro Smart Asset (B.2.2)

Die kummulierten Kosten für das manuelle Bauen von Smart Asset steigen linear, während die kummulierten Kosten für das Bauen von Smart Assets aus der Asset Pipeline konstant bleiben (siehe 5.4).

Für den Smart Asset Build ohne Asset Pipeline ergeben sich konstante Kosten von 473 € pro Smart Asset. Mit der Asset Pipeline kostet ein Smart Asset bei 100 Smart Assets Builds pro Monat 2 € pro Smart Asset (siehe 5.5).

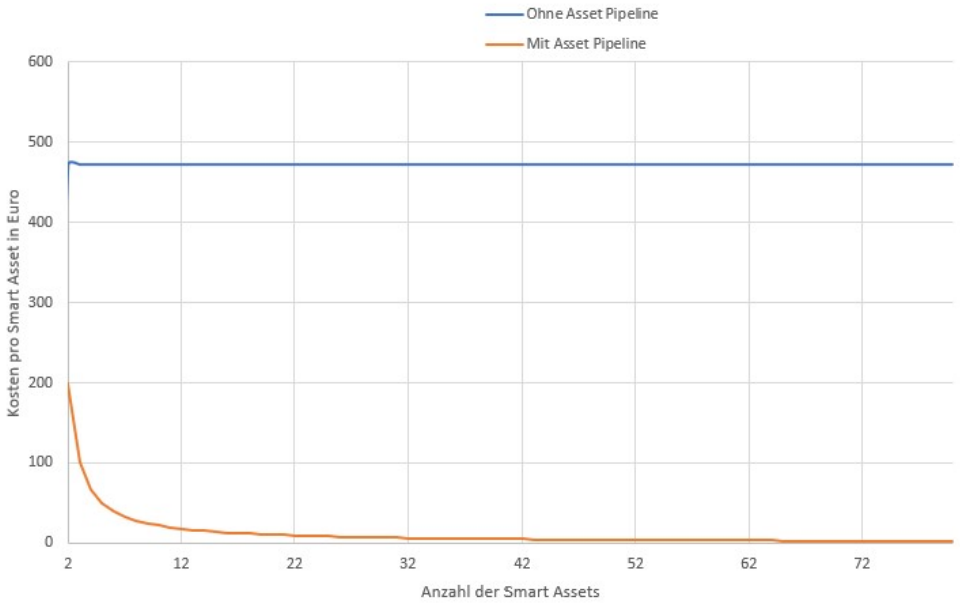


Abbildung 5.5.: Kosten pro Smart Asset bei einem Aufwand von 4,73 Personenstunden pro Smart Asset (B.2.2)

5.3. Adaption von XR

Die Asset Pipeline unterstützt bei der Adaption von XR und Unity in SAP Kundenprojekten, insbesondere im Pre-Sales. SAP bietet in vielen ihrer Produkte kostenlose Trial Zugänge für eine bestimmte Zeit [38]. Dies erlaubt Kunden, unverbindlich in Produkte zu schnuppern. Endet der Trial Zugang, so haben die Kunden die Möglichkeit das Produkt zu kaufen oder darauf zu verzichten.

SAP EPD Kunden, die bislang in die XR Cloud und das Unity Ökosystem reinschnuppern wollen, hatten bislang diese Möglichkeit nicht. Es mussten in jedem Fall Unity Lizenzen gekauft und verwaltet werden, da Unity keine Trial Zugänge anbietet (siehe 2.2.9). Mithilfe der Asset Pipeline fällt die Limitation weg. Es ist nun zum ersten Mal möglich ein Ende-Zu-Ende Trial Zugang für die XR Cloud und XR Viewer Apps aufzubauen, die Pre-Sales Prozesse unterstützen und eine höhere Adaption von XR ermöglichen.

5.4. Limitierungen

Das erfolgreiche Bauen mit der Asset Pipeline wird unter folgenden Bedingungen und Limitierungen unterstützt.

5.4.1. Unity Version

Es werden die Unity Versionen unterstützt, die zum aktuellen Zeitpunkt in den Long-Term-Support Zweigen veröffentlicht werden. Es wurden folgende Versionen erfolgreich getestet.

- 2019.4.20f1 (legacy LTS)
- 2020.3.11f1 (LTS)
- 2021.1.11f1

5.4.2. Render Pipelines

Es werden die bekanntesten drei Rendering Pipelines unterstützt. Hier gibt es jedoch zusätzliche Limitierungen in Hinsicht auf die unterstützten Shader.

- Built-In Render Pipeline: Wird mit allen Standart-Shadern unterstützt.
- Universal Render Pipeline: Wird ausschließlich mit dem "Universal Render Pipeline/Lit"-Shader unterstützt.
- High Definition Render Pipeline: Wird ausschließlich mit dem "HDRP/Lit"-Shader unterstützt.

Modelle, die mit dem SAP EPD Unity Plugin geladen werden besitzen standartmäßig zwei Unity Materialien, die sehr einfach in eine andere Render Pipeline umgewandelt werden. Es werden daher alle EPD Modelle umgewandelt werden können. Die Limitierungen werden erst dann relevant, falls man Modelle auch aus anderen Quellen für die Asset Pipeline nutzen möchte (wie beispielsweise aus Unity Packages).

5.4.3. Zielplattformen

Unterstützte Zielplatformen sind folgende.

- Android
- iOS
- StandaloneWindows64 (Windows)

- StandaloneOSX (MacOS)
- WebGL
- WSAPlayer (Universal Windows Platform)
- StandaloneLinux64

5.4.4. SAP EPD

Es kann pro XR Cloud höchstens ein SAP EPD System, und eine Asset Pipeline genutzt werden. Falls man die Asset Pipeline für mehrere SAP EPD Systeme nutzen möchte, so muss eine weitere XR Cloud Instanz dafür genutzt werden.

5.4.5. Asset Pipeline Status Update

Der Status von Asset Pipeline Aufträgen wird mithilfe eines "Lazy Update" Mechanismus aktualisiert. Es wird bei jedem Aufruf der Smart Asset Version Entität, aber auch beim Aufruf der Smart Asset Pipeline App dieses Update durchgeführt.

Es kann also sein, dass in manchen Fällen das Update länger dauert, falls sehr viele Assets in der Warteschleife waren und die Softwareartefakte große Dateigrößen besitzen. Oftmals ist beim ersten Laden daher die Seite nicht im aktuellsten Zustand. Beim zweiten Laden dagegen wird der aktuelle Zustand wieder angezeigt.

Dieses Problem kann angegangen werden durch das Nutzen des SAP BTP Job Scheduling Services, bei dem der Update Mechanismus in bestimmten Zeitabständen automatisch aufgerufen wird, und nicht über Entitäten. Dies wurde im Rahmen dieser Arbeit aber nicht weiter evaluiert und ist eine potenzielle Erweiterung für die Zukunft.

5.4.6. Netzwerkfehler

In wenigen Fällen bestehen Netzwerkprobleme zwischen der Asset Pipeline und dem SAP EPD. Es kommt daher vor, dass der Downloadprozess nicht erfolgreich ausgeführt werden kann, weil angefragte Daten nicht gesendet wurden. In solchen Fällen wartet die Pipeline auf die Daten und bricht nach maximal einer Stunde ab.

5.5. Testfälle

Die Evaluierung der Implementierung ist durch Testfälle erfolgt. Die Testfälle sollen wieder spiegeln, wie der Endnutzer die Pipeline mit der Pipeline interagiert, und wie die Prozesse zum Erstellen und Nutzen von Smart Assets mit der Asset Pipeline aussehen.

5.5.1. Testfall 1: Konfigurieren der XR Cloud

Der XR Cloud Benutzer mit der Rolle "Admin" muss die XR Cloud zu beginn so konfigurieren, dass die XR Cloud mit dem Azure Devops kommunizieren kann. Des weiteren müssen die nötigen Zugangsdaten für SAP EPD konfiguriert werden, die später zum Laden der Modelle an die Asset Pipeline übergeben werden.

Der Admin geht hierfür auf das Launchpad der XR Cloud und öffnet die App "XR Cloud Konfiguration" (Abb. 5.6).

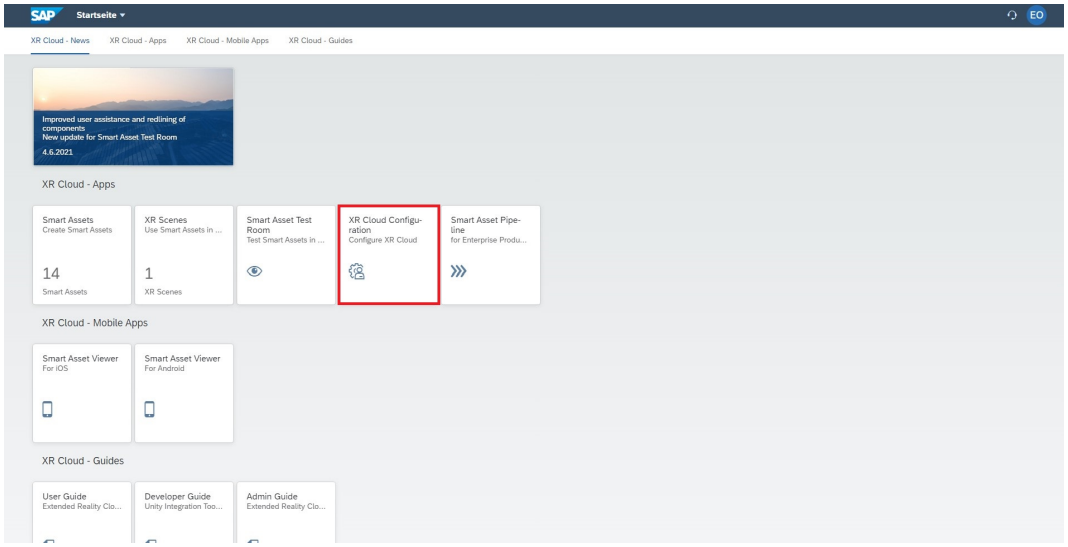


Abbildung 5.6.: Testfall 1: Auswahl der XR Cloud Konfiguration.

In diesem ist der Admin mit Auswahlmöglichkeiten konfrontiert. Es wird als erstes die Asset Pipeline konfiguriert. Dafür klickt der Nutzer auf das Listenelement "Asset Pipeline".

Es erscheint ein Formularfeld, in dem die Zugangsdaten ergänzt werden können. Hat der Benutzer diese ergänzt klickt er auf Speichern und es erscheint eine "Erfolgreich gespeichert"-Meldung (Abb. 5.7).

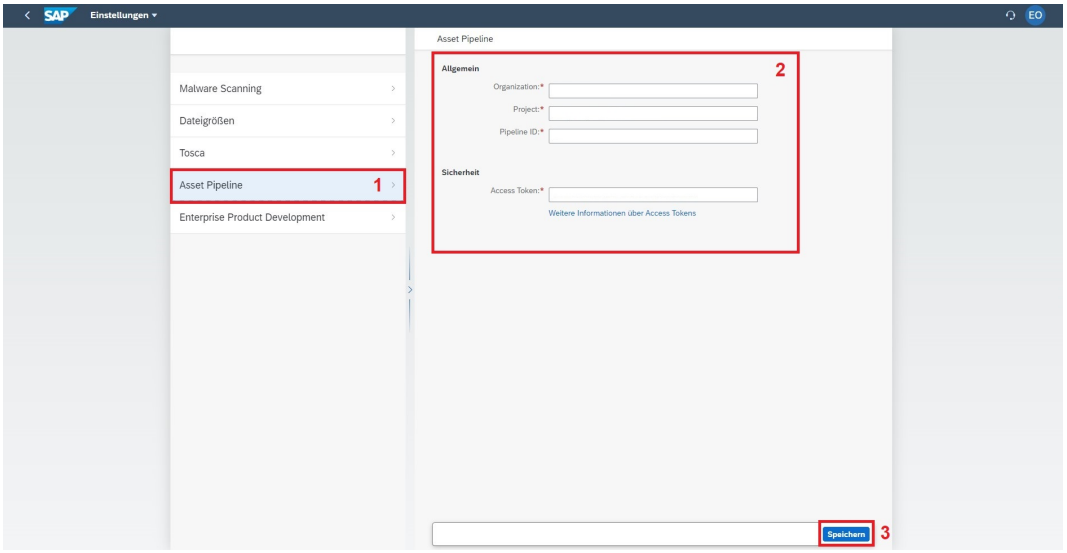


Abbildung 5.7.: Testfall 1: Asset Pipeline Konfiguration.

Im nächsten Schritt konfiguriert der Admin die SAP EPD Zugangsdaten. Dafür klickt er auf das Listenelement "Enterprise Product Development". In dem Formular werden die Zugangsdaten ergänzt, der Admin klickt auf "Speichern" und es erscheint wieder eine "Erfolgreich gespeichert"-Meldung (Abb. 5.8).

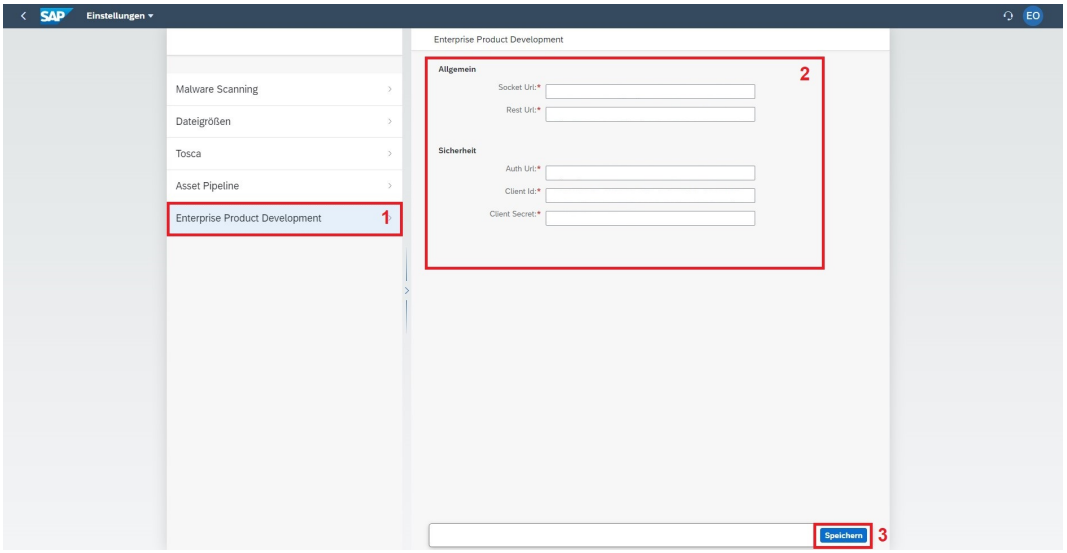


Abbildung 5.8.: Testfall 1: EPD Konfiguration.

5.5.2. Testfall 2: Erstellen von Smart Assets

Im nächsten Schritt kommt der Benutzer mit der Rolle "Creator" ins Spiel. Der XR Cloud Benutzer mit dieser Rolle muss ein Smart Asset erzeugen und dieses Smart Asset so konfigurieren, dass sie von der Asset Pipeline genutzt werden kann. Dafür öffnet er die Smart Assets App (Abb. 5.9).

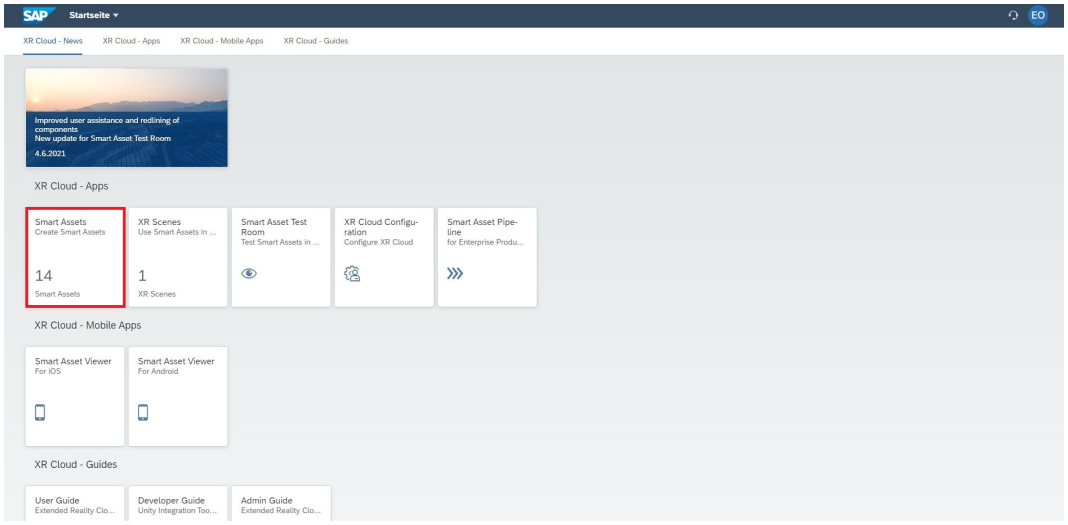


Abbildung 5.9.: Testfall 2: Auswahl der Smart Assets App.

Er wählt "Anlegen" um ein Smart Asset zu erzeugen. Es öffnet sich ein neues Fensterbereich, in dem die Informationen zum Smart Asset ergänzt werden können (Abb. 5.11). Es wird ein Name und eine Beschreibung angegeben. Es wird der "Teilen" Button betätigt, damit das Smart Asset öffentlich ist und später auch im Smart Asset Test Room eingebunden werden kann.

Ist dies geschehen, wird zum Smart Asset eine neue Version angelegt. Eine Version dient zur Unterscheidung verschiedener Konfigurationen des Modells. Hierfür scrollt der Benutzer runter in den "Versionen"-Bereich und betätigt den "Anlegen"-Button.

Im Feld Version wird die Bezeichnung der Version angegeben. Anschließend werden die EPD Parameter angegeben. Die EPD Parameter geben an, welches Modell von der Asset Pipeline geladen werden soll und wie dieses Modell konfiguriert werden soll. Am Wichtigsten ist hierbei das Feld SScene ID", welches die ID des Modelles in SAP EPD angibt. In den Deteils in SAP EPD kann dieser Wert bei jedem Modell angefragt werden (siehe 5.10). In diesem Testfall werden die Work Instructions (Animationen) und Metadaten aktiviert. Die Skalierung des Modells wird angepasst, da das Modell in SAP EPD eine andere Maßeinheit nutzt als Unity.

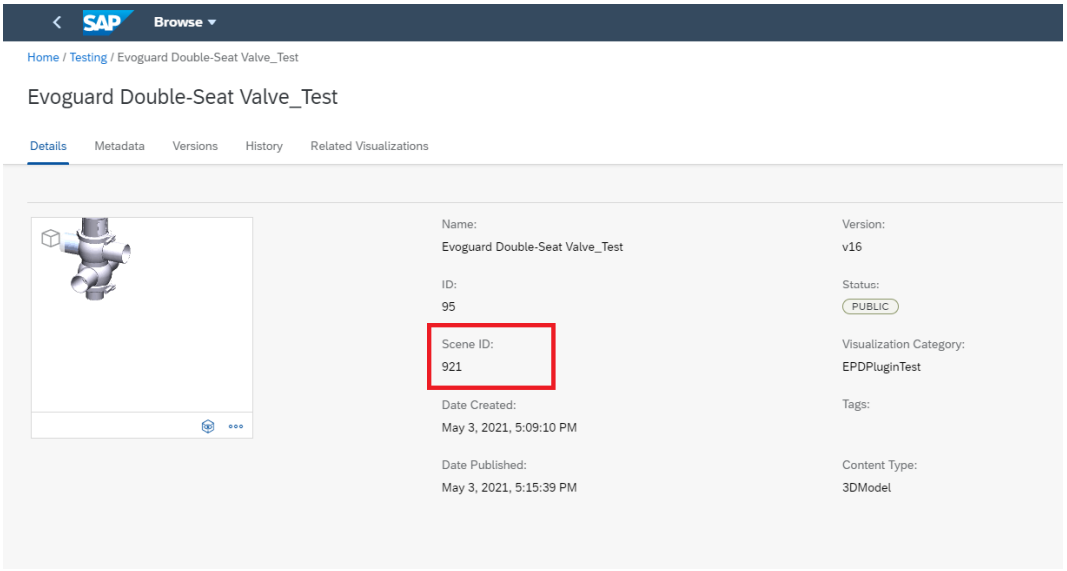


Abbildung 5.10.: Details eines SAP EPD Modells

Zum Abschluss betätigt der Benutzer den "Übernehmen"-Button zum Speichern der Version und ÄnlegenButton zum Anlegen des Smart Assets. Damit wurde ein Smart Asset zur weiteren Benutzung in der Pipeline konfiguriert.

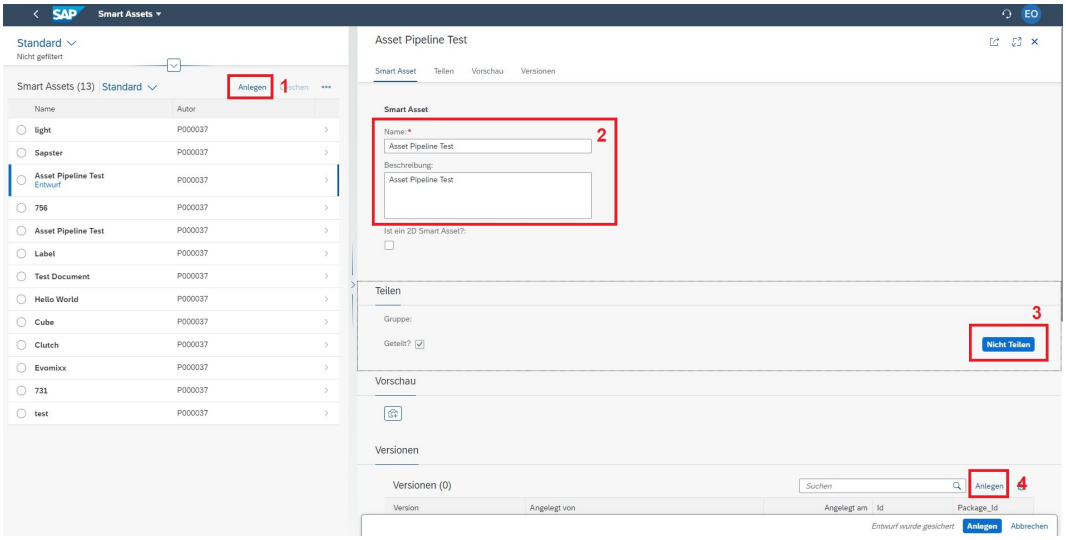


Abbildung 5.11.: Testfall 2: Erstellen eines neuen Smart Assets.

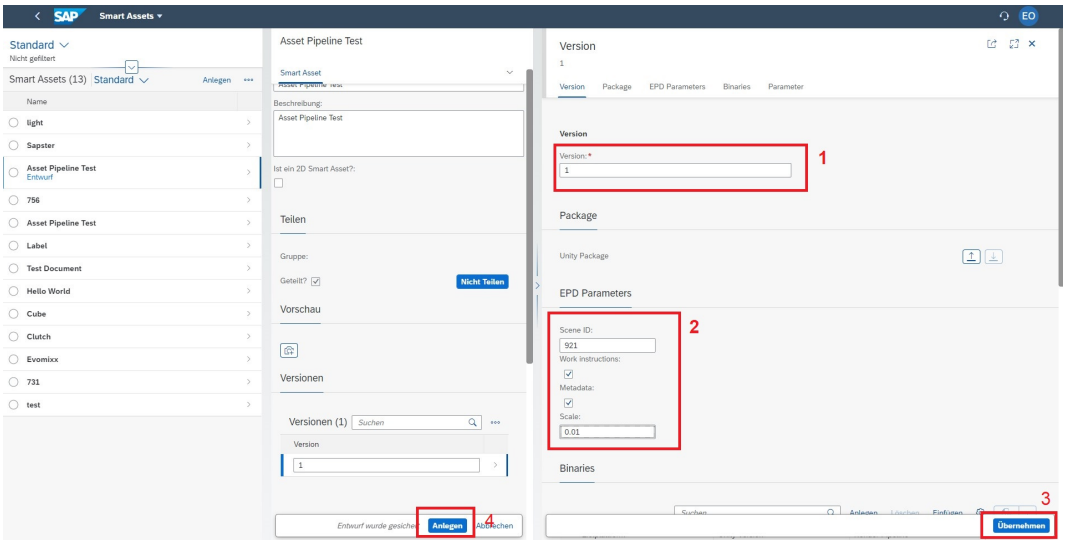


Abbildung 5.12.: Testfall 2: Erstellen eine Smart Asset Version.

5.5.3. Testfall 3: Das Smart Asset im Smart Asset Test Room darstellen

Ist das Smart Asset konfiguriert, kann der Benutzer mit der Rolle Üser"nun das Smart Asset in jeder beliebigen XR Cloud Szene nutzen und dafür die Smart Asset Binaries bauen lassen. Der Benutzer öffnet die "XR Scenes"-App (Abb. 5.13).

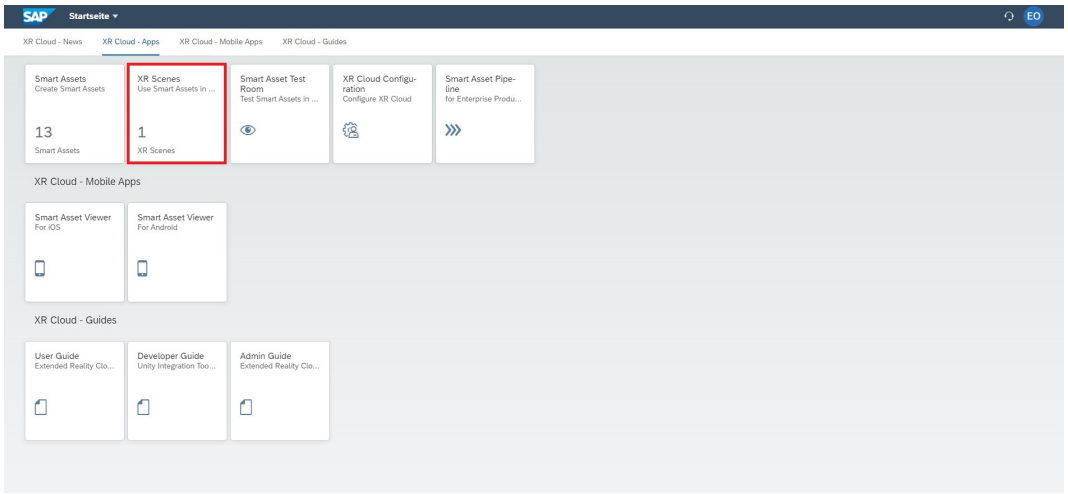


Abbildung 5.13.: Testfall 3: Auswahl der XR Scenes App.

Es erscheint die Auflistung der vorhandenen XR Cloud Szenen. Es wird die SSmart Asset Test RoomSzene geöffnet, indem der Pfeil in der jeweiligen Zeile betätigt wird. Es öffnen sich die Smart Assets in dieser Szene (Abb. 5.14).

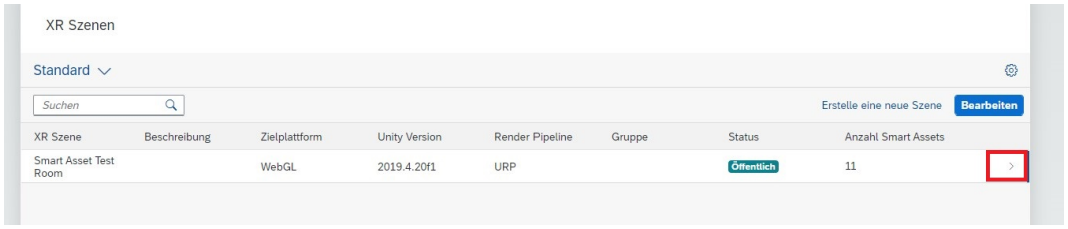


Abbildung 5.14.: Testfall 3: Auflistung der XR Cloud Szenen.

Im Kopfbereich der Szene ist die Konfiguration der Szene einsehbar. Hier kann durch das Betätigen des "Füge Smart Asset zur Szene HinzuButtons ein neuer Eintrag ergänzt werden (Abb. 5.15).

Durch das Klicken des "Smart Asset"-Feldes kann die Suchhilfe geöffnet werden (Abb. 5.16). In dieser Suchhilfe wird das Feld für die neueste Version ausgeblendet, damit alle vorhandenen Smart Assets angezeigt werden. Der Benutzer sucht das Erzeugte Smart Asset in der Liste. Da das Asset neu erzeugt wurde, existiert noch keine Binary. Durch das Klicken des "Binary bauenButtons kann die Pipeline

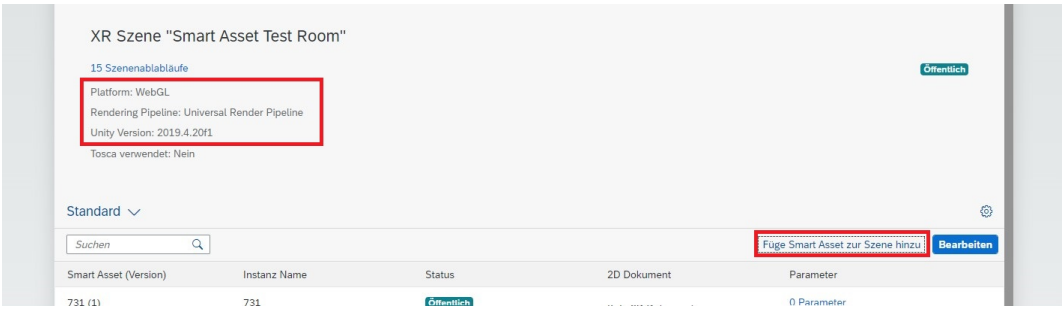


Abbildung 5.15.: Testfall 3: XR Cloud Szene Smart Asset Test Room.

angestoßen werden. Der Benutzer kann anschließend durch das Klicken des Smart Asset Namens das Smart Asset bereits in die Szene hinzufügen.

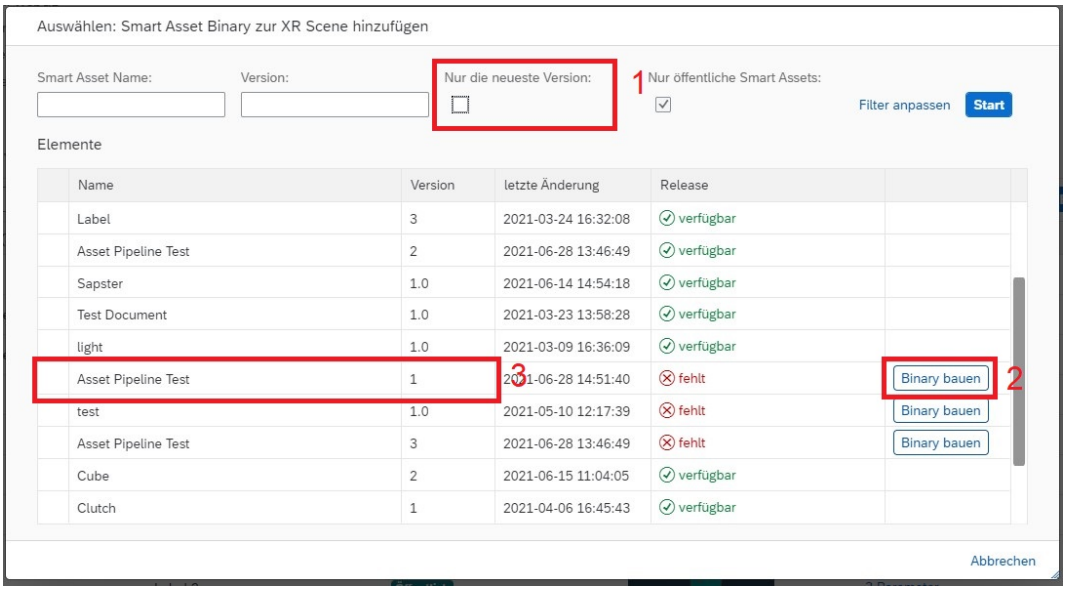


Abbildung 5.16.: Testfall 3: Suchhilfe für Smart Assets.

Es ist nun möglich den Status des Pipeline Jobs einzusehen. Hierfür wird wieder das Launchpad geöffnet. Im Launchpad wird die Smart Asset Pipeline App geöffnet (Abb. 5.17).

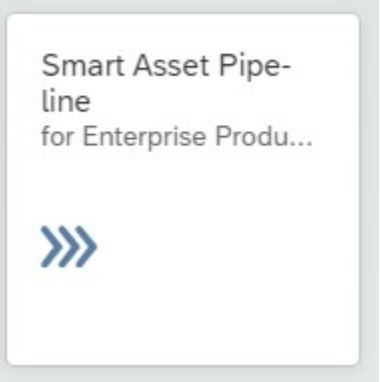


Abbildung 5.17.: Testfall 3: Smart Asset Pipeline App.

Hier ist der Eintrag zu sehen (Abb. 5.18). Der Pipeline Job sollte 5-10 Minuten dauern. Nach Abschluss wird der Status von "Laufend" zu "Abgeschlossen" geändert. Nun ist das Smart Asset fertig gebaut worden.

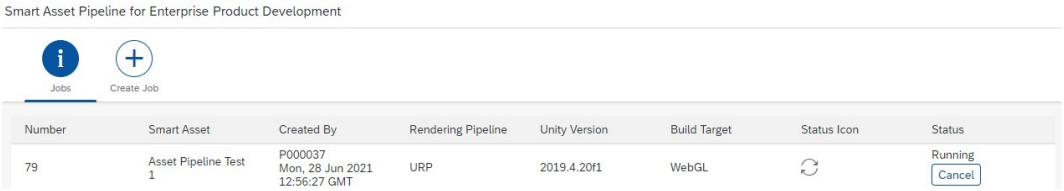


Abbildung 5.18.: Testfall 3: Pipeline Einträge.

Der Benutzer geht nun wieder aufs Launchpad und öffnet die Smart Asset Test Room App. Mit dieser App kann das erzeugte Smart Asset getestet und eingesehen werden. Ist das Modell ausgewählt erscheint am rechten Rand die Metadatenleiste. Beim Klicken auf bestimmte Teile sind die Metadaten dieser Teile sichtbar. Der Benutzer kann auf den "Animations" Button klicken, um die Steuerung der Animationen zu öffnen (Abb. 5.19).

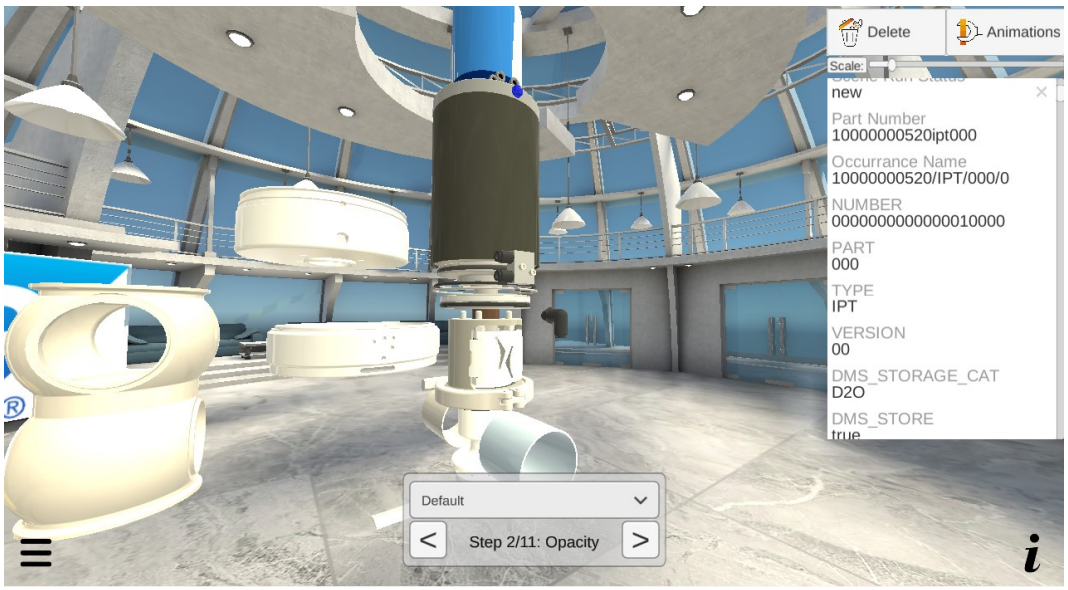


Abbildung 5.19.: Testfall 3: Dargestelltes Smart Asset im Smart Asset Test Room inklusive Metadaten und Animationen.

Damit wird der Testfall abgeschlossen.

5.5.4. Testfall 4: Mehrere Smart Assets als Batch-Job bauen

Es kommt vor, dass ein bestimmtes Smart Asset für eine größere Menge an Kombinationen an Unity Versionen, Render Pipelines und Zielplattformen gebaut werden muss. Zum Durchführen einer solchen Operation öffnet der Benutzer mit der Rolle "User" im Launchpad der XR Cloud die "Smart Asset Pipeline" App (Abb. 5.20).

Im oberen Bereich der App wechselt der Benutzer auf den "Create Job"-Tab. Es öffnet sich ein Formularfeld (Abb. 5.21). In diesem Formularfeld gibt es vier Felder.

- Smart Asset: Es wird das Smart Asset gewählt, was in Testfall 2 erzeugt wurde, also Asset Pipeline Test Version 1.

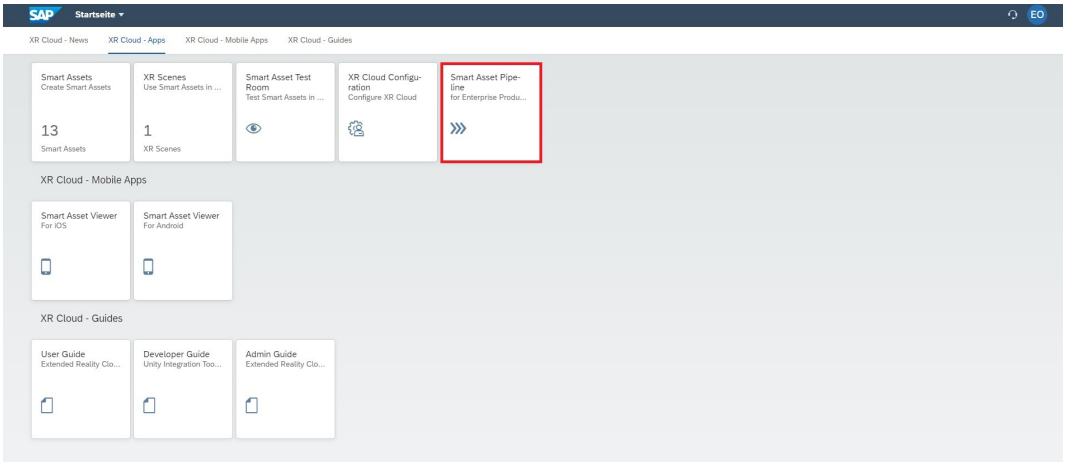


Abbildung 5.20.: Testfall 4: Smart Asset Pipeline App.

- Unity Version: Durch das Klicken auf das Icon für die Suchhilfe öffnet sich die Menge Auswählbarer Unity Versionen. Der Benutzer wählt hier 2019.4.20f1. Er will jedoch weitere Unity Versionen hinzufügen, die nicht in der Suchhilfe sind. Dafür klickt er auf das Eingabefeld. Der Cursor der Tastatur erscheint dort und es kann eine manuelle Eingabe getätigt werden. Der Benutzer gibt nun "2020.3.11f1" und "2021.1.11f1" und betätigt jeweils die "Enter"-Taste.
- Rendering Pipleine: Es wird "URP", "HDRP" und "BuiltIn" über die Suchhilfe ausgewählt.
- Build Target: Es werden die Zielplattformen "Android", "iOS", "WSAPlayer", "StandaloneWindows64", "StandaloneOSX" und "WebGL" ausgewählt.

Das Klicken auf den "Absenden"-Button führt dazu, dass nun Pipeline Aufträge in den verschiedenen Kombinationen ausgeführt werden.

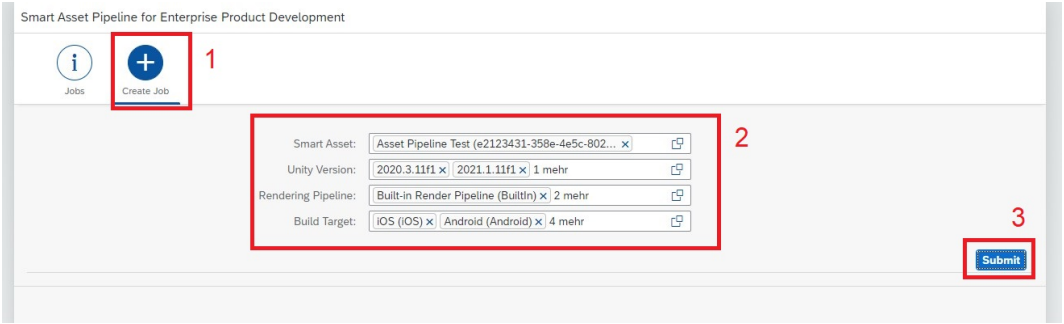
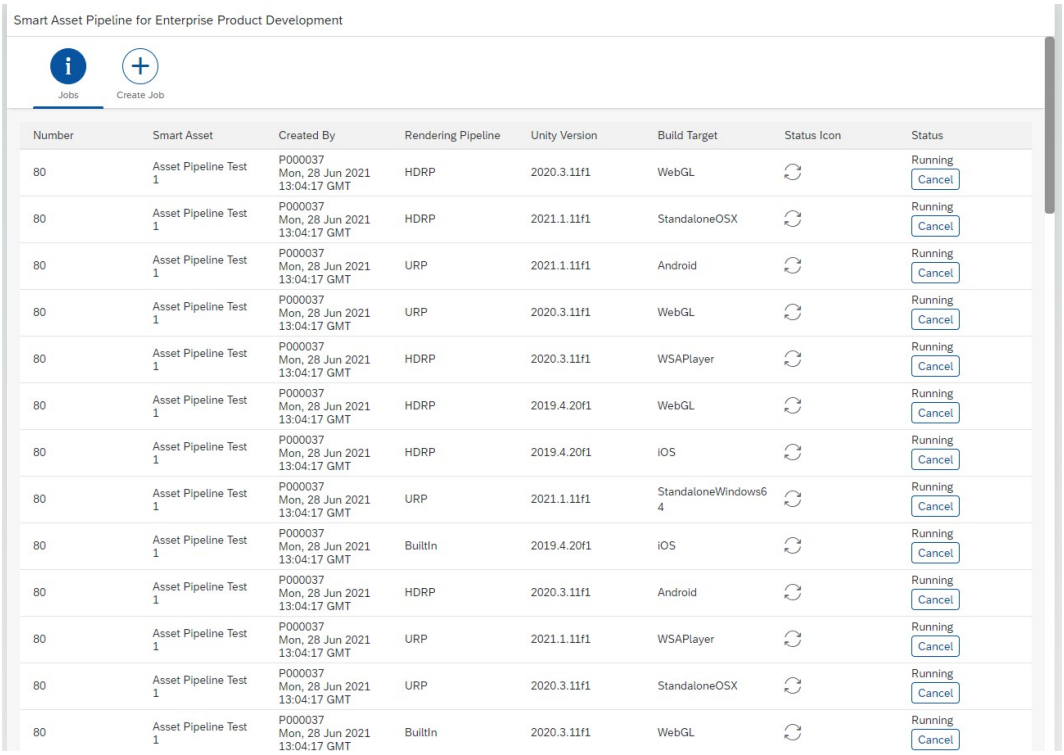


Abbildung 5.21.: Testfall 4: Erstellen von mehreren Pipeline Aufgaben.

Nach dem Aktualisieren der aktuellen Seite, ist wieder der "Jobs"-Tab ausgewählt. Hier sind nun alle erstellen Pipeline Jobs einsehbar (Abb. 5.22).



Number	Smart Asset	Created By	Rendering Pipeline	Unity Version	Build Target	Status Icon	Status
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	HDRP	2020.3.11f1	WebGL		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	HDRP	2021.1.11f1	StandaloneOSX		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	URP	2021.1.11f1	Android		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	URP	2020.3.11f1	WebGL		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	HDRP	2020.3.11f1	WSAPlayer		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	HDRP	2019.4.20f1	WebGL		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	HDRP	2019.4.20f1	iOS		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	URP	2021.1.11f1	StandaloneWindows64		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	BuiltIn	2019.4.20f1	iOS		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	HDRP	2020.3.11f1	Android		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	URP	2021.1.11f1	WSAPlayer		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	URP	2020.3.11f1	StandaloneOSX		Running Cancel
80	Asset Pipeline Test 1	P000037 Mon, 28 Jun 2021 13:04:17 GMT	BuiltIn	2020.3.11f1	WebGL		Running Cancel

Abbildung 5.22.: Testfall 4: Auflistung der neuen Pipeline Aufgaben in der Smart Asset Pipeline App.

5.5.5. Ergebnis

Für die Durchführung der Tests wurde Jira genutzt. Die Tests wurden von drei Mitgliedern des XR Venture Teams durchgeführt. Die Ergebnisse der Tests sind in B.1 zu finden. Bei den Tests sind Verbesserungsvorschläge entstanden, die als Backlog-Item für zukünftige Arbeiten dokumentiert wurden. Abgesehen davon wurden alle Tests erfolgreich durchgeführt und die Implementierung wurde vom XR Venture Team abgenommen.

Kapitel 6

Fazit

6.1. Zusammenfassung

SAP Kunden bringen spezielle Anforderungen mit, wenn es darum geht 3D Modelle mit ERP Bezug in 3D Anwendungen darzustellen. Das Smart Asset Konzept verspricht, diese Anforderungen zu erfüllen, um kommerzielle Softwarelösungen in Extended Reality zu entwickeln. 3D Modelle können als Smart Assets schneller in eine 3D Anwendung geladen werden, als beispielsweise direkt über existierende Schnittstellen von SAP Systemen. Dies wurde mit Benchmarks nachgewiesen (5.1).

Smart Assets bringen jedoch einen entscheidenden Nachteil. Smart Assets sind Binärdateien, die abhängig von Unity Version, Zielplattform und Render Pipeline sind. Ein Unity Entwickler muss geschult sein, um funktionsfähige Smart Assets zu bauen. Es war bislang ein manueller Prozess mit einem hohen Arbeitsaufwand. Dieser Aufwand wurde im Rahmen dieser Arbeit mit 4,73 Personenstunden pro Smart Asset abgeschätzt (5.7). Dieser kann ein Hindernis für die Adaption von Smart Assets in Kundenprojekten darstellen.

Eine Asset Pipeline kann den Build Prozess für Smart Assets skalieren. Im Rahmen dieser Arbeit wurde zum ersten Mal eine solche Asset Pipeline eigenständig entwickelt. Die Asset Pipeline integriert sich in SAP EPD, XR Cloud und Azure Devops. Es lässt sich aus beliebigen SAP EPD Modellen Smart Assets bauen. In dieser Arbeit wurde herausgefunden, dass die Kosten pro Smart Asset mit der Asset Pipeline mit jedem weiteren Smart Asset linear sinken. Das Betreiben der Asset Pipeline kostet monatlich 200 €. Bei 100 Smart Assets pro Monat beträgt der Preis pro Smart Asset somit 2€. Zum Vergleich beträgt im manuellen Smart Asset Build

Prozess der Preis konstant 473 € pro Smart Asset. Der Break-Even-Point für die Asset Pipeline ist dadurch bereits beim ersten Build (5.5).

Die Asset Pipeline unterstützt die Kommerzialisierung von Smart Assets und Extended Reality in Pre-Sales Prozessen. SAP Kunden, die in die Technologie reinschnuppern möchten, mussten bisher eigene Unity Entwickler einstellen, um Smart Assets bauen zu können. Mit der Asset Pipeline können SAP EPD Kunden ihre Modelle in Extended Reality Viewern darstellen, ohne eigene Entwickler einzustellen. Zur Generierung von Leads kann mithilfe der Asset Pipeline zum ersten Mal ein Ende-Zu-Ende Trialzugang für die Darstellung von Smart Assets in Extended Reality aufgebaut werden (5.3).

6.2. Diskussion

6.2.1. Unity Integration Toolkit

Die Änderungen am UIT betreffen das Smart Asset Unity Plugin und das SAP EPD Unity Plugin. Es wurde ein Review-Prozess durchgeführt, um die Änderungen zu prüfen. Dank dieser Änderungen wurde die Nutzung der Plugins für die Smart Asset Pipeline ermöglicht.

Im Rahmen der Evaluierung wurden weitere Punkte gefunden, an denen Verbesserungen durchgeführt werden können. Dies betrifft unter anderem die Fehlerbehandlung. Fehler, die im SAP EPD Plugin passieren, können über SAP EPD Schnittstellen nur bedingt abgefangen werden. Dies betrifft unter anderem die Eingabe falscher Zugangsdaten oder Parameter. Für die Asset Pipeline wurde eine unsaubere Implementierung in dieser Hinsicht durchgeführt, um Fehler abzufangen.

In Zukünftigen Arbeiten muss die Fehlerbehandlung im SAP EPD Unity Plugin verbessert werden, damit in Asset Pipeline Jobs bessere Fehlermeldungen ausgegeben werden können.

6.2.2. Compliance

Im Umfeld der Unternehmenssoftware ist Sicherheit und Compliance kritisch. Im Rahmen von früheren Projekten wurde festgestellt, dass das Installieren von Unity aus Downloads der offiziellen Webseite nicht compliant ist, weil die Downloaddatei

nicht geprüft werden kann. Aus diesem Grund war es bisher nicht möglich, dynamisch eine Unity Version in bestehenden Pipelines des XR Ventures zu installieren. In dieser Arbeit wurde zum ersten Mal Unity Hub zum Installieren von Unity genutzt. Es ist der offizielle von Unity angebotene Weg, um weitere Unity Versionen zu installieren. Eine erneute Evaluierung der Compliance steht aus.

6.2.3. Render Pipelines

Die Asset Pipeline unterstützt alle Modelle aus SAP EPD. Das SAP EPD Unity Plugin besitzt zwei einfache Materialien, bei der eine Umwandlung der Shader möglich ist. Falls die Asset Pipeline so erweitert werden sollte, dass Modelle auch aus anderen Quellen (z.B. Packages) kommen können, so müssen die Render Pipeline Umwandlungen näher untersucht werden. Die Asset Pipeline unterstützt bislang eine Basisfunktionalität für die Umwandlung von Shadern unter verschiedenenen Render Pipelines.

6.3. Ausblick

Zeitgleich zu dieser Arbeit sind fast alle im XR Venture Team damit beschäftigt, die AR und MR Viewer Apps für Hololens, iOS und Android aufzubauen. Diese Apps werden ab August weltweit für alle SAP EPD Kunden ausgerollt.

Die Inhalte dieser Apps kommen mithilfe des SAP EPD Unity Plugins direkt aus SAP EPD. Als langfristige Erweiterung steht zur Diskussion, dass die Inhalte aus der XR Cloud kommen und die Vorteile des Smart Asset Konzepts hier genutzt werden. Die Asset Pipeline wird hierbei eine wichtige Rolle spielen: Es wird nötig sein, diese Smart Assets automatisiert für die verschiedenen Plattformen (Hololens, iOS, Android) und Unity Versionen zu bauen. Die XR Viewer Apps sind der erste Anwendungsfall, in dem die Asset Pipeline zur Nutzung kommen könnten.

6.4. Danksagung

Es wäre ignorant zu sagen, dass das einzig meine eigene Arbeit ist. Diese Arbeit wäre nicht ohne die Unterstützung unzähliger Personen zustande gekommen. Hier möchte ich die Gelegenheit nutzen, diesen mein Dank auszusprechen.

Der größte Dank geht an Michael Spieß. Michael hat jedem von uns im XR Venture Team von Tag 1 an Aufgaben gegeben, die Verantwortung und Zweck mit bringen. Als Erfinder, Architekt und treibende Kraft für XR in SAP ist er für uns alle ein Vorbild. In seinem Team habe ich durchgehend das Gefühl, dass ich als Entwickler tatsächlich mit daran beteiligt bin, die Zukunft für Unternehmenssoftware mitzuformen.

Ich möchte einen großen Dank an Prof. Frank Dopatka aussprechen. Zum ersten Mal mit Unity in Kontakt kam ich dank seiner Game-Engineering Vorlesung. Es handelt sich um keine klassische Vorlesung, mehr eine innovative Veranstaltung, die sich besonders durch das Selbststudium und die Selbstverpflichtung der Studierenden auszeichnet.

Ich möchte mich beim ganzen XR Venture Team bedanken. Hierbei möchte ich mich namentlich bei Moritz-Peter Valerius, Jonas Lipps, Maksim Pahlberg, David Schrott, Michael Spieß bedanken. Der Innovationsgeist in diesem Team ist etwas, was ich sonst nirgendwo wahr genommen habe. Es ist mir eine Ehre, Teil dieses Teams zu sein.

Inbesondere möchte ich mich hierbei bei Maksim Pahlberg und David Schrott bedanken. Sie haben mich während dieser Arbeit mit ihrer außergewöhnlichen technischen Expertise unterstützt. Die Asset Pipeline ist dank ihrer Unterstützung in kürzester Zeit zu einer (meiner Meinung nach) hohen Reife gelangt.

Ich möchte meiner Frau, Meryem Tamer-Ördek, dafür danken, dass sie mich immer dabei unterstützt noch besser zu werden. Ich danke ihr, dass sie während meiner Bachelorarbeit Rücksicht gezeigt hat.

Ich möchte meinen Eltern, Yüksel und Gültekin Ördek, dafür danken, dass Sie immer an mich geglaubt haben. Sie haben bereits sehr früh in die Bildung ihrer Kinder investiert, teilweise ohne selber einen Schulabschluss zu haben.

Diese Arbeit wurde durch die SAP SE finanziert. Ich hatte die Ehre, in den letzten zwei Jahren mit großartigen Herausforderungen und kompetenten Menschen arbeiten zu dürfen. Der beste Arbeitgeber der Welt soll auch die beste Arbeit der Welt

bekommen. Auch wenn diese Arbeit diesem Anspruch nicht gerecht wird, hoffe ich, dass ich meinen Beitrag darin geleistet habe, die Welt gemäß der Mission "Helping the World run better" ein klein bisschen was beizutragen.

Ich danke dir mein Leser, dass du bis hierhin gelesen hast.

Abkürzungsverzeichnis

UIT Unity Integration Toolkit
CI Continuous Integration
XR Extended Reality
VR Virtual Reality
MR Mixed Reality
AR Augmented Reality
HDRP High Definition Render Pipeline
URP Universal Render Pipeline
SRP Scriptable Render Pipeline
DLC Downloadable Content
CAD Computer Aided Design
CTO Chief Technology Officer
LTS Long Term Support
EPD Enterprise Product Development
BTP Business Technology Platform
VM Virtuelle Maschine
UWP Universal Windows Platform
ERP Enterprise Resource Planning
REST Representational State Transfer

Tabellenverzeichnis

Abbildungsverzeichnis

Listings

Literatur

[1] Ralf Kreutzer und Karl-Heinz Land, *Digitaler Darwinismus*. Jan. 2016. DOI: 10.1007/978-3-658-11306-3. Adresse: http://dx.doi.org/10.1007/978-3-658-11306-3.

[2] R. Asghari R. Schlimbach. (2020). „Das Digital Canvas: Ein Instrument zur Konzeption digitaler Geschäftsmodelle“, Adresse: https://doi.org/10.1365/s40702-020-00624-9 (besucht am 11. 11. 2020).

[3] Christian Klein. (2021). „Introducing RISE with SAP – Business Transformation as a Service“, Adresse: https://news.sap.com/2021/01/rise-with-sap-business-transformation-as-a-service/ (besucht am 22. 03. 2021).

[4] Manuel Brunner und Josef Wolfartsberger, „Virtual Reality enriched Business Model Canvas Building Blocks for enhancing Customer Retention“, *Procedia Manufacturing*, Jg. 42, S. 154–157, 2020, International Conference on Industry 4.0 and Smart Manufacturing (ISM 2019). DOI: https://doi.org/10.1016/j.promfg.2020.02.062. Adresse: https://www.sciencedirect.com/science/article/pii/S2351978920306260.

[5] „AR and MR Viewers for SAP Enterprise Product Development“, 2021. Adresse: https://www.sap.com/dmc/exp/2021-06-sapphireinnovation-news/ar-and-mr-viewers-for-sap-enterprise-product-development/ (besucht am 13. 07. 2021).

[6] „Revolution durch XR Process Experience – Mitarbeitende werden zu Helden der Digitalen Transformation und gestalten Prozesse virtuell mit“, 2021. Adresse: https://www.bearingpoint.com/de-de/ueber-uns/pressemitteilungen-und-medienberichte/pressemitteilungen/revolution-durch-xr-process-experience/ (besucht am 13. 07. 2021).

[7] Alan R. Hevner, Salvatore T. March, Jinsoo Park und Sudha Ram, „Design Science in Information Systems Research“, *MIS Quarterly*, Jg. 28, Nr. 1, S. 75–105, 2004. Adresse: http://doi.org/10.2307/25148625.

[8] Thomas P Kersten, Felix Tschirschwitz, Simon Deggim und Maren Lindstaedt, „Virtual Reality–Von der 3D-Erfassung bis zum immersiven Erlebnis“, 2018. Adresse: https://tubaf.qucosa.de/id/qucosa:23241.

[9] Ronald T. Azuma, „A Survey of Augmented Reality“, 1997. Adresse: https://doi.org/10.1162/pres.1997.6.4.355.

[10] Maximilian Speicher, Brian D. Hall und Michael Nebeling, „What is Mixed Reality?“, in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2019, S. 1–15. Adresse: https://doi.org/10.1145/3290605.3300767.

[11] „Was ist Mixed Reality?“, Adresse: https://docs.microsoft.com/de-de/windows/mixed-reality/discover/mixed-reality (besucht am 13. 07. 2021).

[12] „SAP Future Office, Mixed Reality Team“, Adresse: https://www.youtube.com/watch?v=KsCaZtk87ok (besucht am 13. 07. 2021).

[13] Sebastian Friston, Carmen Fan, Jozef Doboš, Timothy Scully und Anthony Steed, „3DRepo4Unity: Dynamic Loading of Version Controlled 3D Assets into the Unity Game Engine“, in *Proceedings of the 22nd International Conference on 3D Web Technology*, Ser. Web3D ’17, Brisbane, Queensland, Australia: Association for Computing Machinery, 2017. DOI: 10.1145/3055624.3075941. Adresse: https://doi.org/10.1145/3055624.3075941.

[14] „Most used libraries, frameworks, and tools among developers, worldwide, as of early 2020“, Adresse: https://www.statista.com/statistics/793840/worldwide-developer-survey-most-used-frameworks/ (besucht am 13. 07. 2021).

[15] „Unity LTS“, Adresse: https://unity3d.com/de/unity/qa/lts-releases (besucht am 13. 07. 2021).

[16] „Build once, deploy anywhere“, Adresse: https://unity.com/features/multiplatform (besucht am 13. 07. 2021).

[17] „Important Classes - GameObject“, Adresse: https://docs.unity3d.com/Manual/class-GameObject.html (besucht am 13. 07. 2021).

[18] „Render pipelines“, Adresse: https://docs.unity3d.com/Manual/render-pipelines.html (besucht am 13. 07. 2021).

[19] „AssetBundles“, Adresse: https://docs.unity3d.com/Manual/AssetBundlesIntro.html (besucht am 13. 07. 2021).

[20] „Creating custom packages“, Adresse: https://docs.unity3d.com/Manual/CustomPackages.html (besucht am 13. 07. 2021).

[21] „Unity's Package Manager“, Adresse: https://docs.unity3d.com/Manual/Packages.html (besucht am 13. 07. 2021).

[22] „The details of entering Play Mode“, Adresse: https://docs.unity3d.com/2019.3/Documentation/Manual/ConfigurableEnterPlayModeDetails.html (besucht am 13. 07. 2021).

[23] „Order of execution for event functions“, Adresse: https://docs.unity3d.com/Manual/ExecutionOrder.html (besucht am 13. 07. 2021).

[24] „Unity Terms of Service“, Adresse: https://unity3d.com/legal/terms-of-service (besucht am 13. 07. 2021).

[25] „SAP Enterprise Product Development“, Adresse: https://www.sap.com/products/enterprise-product-development.html (besucht am 13. 07. 2021).

[26] XR Venture Team, „Unity Integration Toolkit Developer Guide“, 2020. Adresse: https://xrcloud-docs.cfapps.eu10.hana.ondemand.com/developerguide-uit/.

[27] Paul Lilly, „Oculus Quest is giving VR the boost it needs for mass adoption“, Adresse: https://www.pcgamer.com/au/oculus-quest-is-giving-vr-the-boost-it-needs-for-mass-adoption (besucht am 13. 07. 2021).

[28] „Choose the plan that is right for you“, Adresse: https://store.unity.com/compare-plans (besucht am 13. 07. 2021).

[29] XR Venture Team, „Extended Reality Cloud User Guide“, 2020. Adresse: https://xrcloud-docs.cfapps.eu10.hana.ondemand.com/userguide-xrcloud/.

[30] „SAP 3D Visual Enterprise“, Adresse: https://www.sap.com/products/product-visualization.html (besucht am 13. 07. 2021).

[31] „Azure DevOps Services REST API Reference“, Adresse: https://docs.microsoft.com/en-us/rest/api/azure/devops/?view=azure-devops-rest-6.1 (besucht am 13. 07. 2021).

[32] „Segmentation Fault with Win32API“, Adresse: https://github.com/DragonBox/u3d/issues/414 (besucht am 13. 07. 2021).

[33] „Installing Unity“, Adresse: https://docs.unity3d.com/Manual/GettingStartedInstallingUnity.html (besucht am 13. 07. 2021).

[34] „DragonBox/u3d“, Adresse: https://github.com/DragonBox/u3d (besucht am 13. 07. 2021).

[35] „Self-hosted Windows agents“, Adresse: https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops (besucht am 13. 07. 2021).

[36] „Windows 10 SDK“, Adresse: https://developer.microsoft.com/de-de/windows/downloads/windows-10-sdk/ (besucht am 13. 07. 2021).

[37] „Vodafone Produktinformationen eazy40“, Adresse: https://eazy.de/pdf/vertriebswerk/PIB_eazy40.pdf (besucht am 13. 07. 2021).

[38] „SAP Software Trials“, Adresse: https://www.sap.com/products/free-trials.html (besucht am 13. 07. 2021).

Anhang A

Erster Anhang

A.0.1. Interview mit Experte A

Angaben zum Interview:

- **Interviewpartner**: Experte A
- **Rolle**: Unity Entwickler, Cloud Entwickler, UI5 Entwickler
- **Datum**: 18.06.2021, 13:30 – 14:00
- **Ort**: Microsoft Teams Meeting
- **Durchgeführt von**: Enes Ördek
- **Einverständniserklärung**: Ja
- **Typ**: mündlich
- **Transkripiert von**: Enes Ördek

Einstiegsfragen

Wie lange arbeitest du bereits mit Unity?

Mit Unity sind es jetzt 14 Jahre. Seit 2007 müsste das sein, als die erste freie Unity Version raus gekommen ist.

Was hast du bereits mit Smart Assets /Asset Bundles gearbeitet?

Ich habe Smart Assets im Rahmen der SAP Tätigkeit benutzt. Asset Bundles habe ich immer wieder bei meinen Projekten verwendet. Ich habe sie für das dynamische Laden von GameObjects benutzt.

Folgefrage: Mit Addressable Assets auch?

Addressables habe ich noch nicht benutzt, aber Addressables nutzen ja auch Asset Bundles.

Schlüsselfragen

Was stellst du dir unter einer Asset Pipeline allgemein vor?

Eine Asset Pipeline sollte möglichst automatisiert und mit wenig Eingriff Assets erzeugen können für verschiedene Zielplattformen.

Welche Probleme gibt es aktuell und welche Probleme soll die Asset Pipeline lösen?

Wenn man Modelle aus einem 3rd Party System wie EPD in 3D Umgebungen wie Unity verwenden möchte, muss man diese vorher importieren und bauen.

Das ist immer sehr viel manueller Aufwand, der entsteht. Für jede Unity Version und für jede Zielplattform müsste dieser Prozess durchgeführt werden.

Welche Eingangsdaten hat die Asset Pipeline? EPD Parameter? Package? Konfiguration?

Es kommt darauf an, wie man das Ganze abstrahiert. Wenn wir ausgehen, dass wir bereits ein Gameobjekt haben, dann haben wir Parameter wie Unity Version, Build Target – aber auch für das Gameobjekt selber sowas wie Skalierung. Eventuell auch Position und Rotation, wobei das eher weniger relevant wäre. Zusätzlich noch die Rendering Pipeline.

Dann ist noch die Frage, wo kommt das Gameobjekt her. Optimalerweise haben noch wir gar kein Gameobjekt, sondern sparen uns den Unity teil komplett. Dann wäre der Parameter, aus welchem System das kommt, und was die ID ist. Zusätzlich können Parameter in Frage kommen, die es in Unity noch nicht gibt, aber die im anderen System vorhanden sind.

Was ist die Ausgabe der Asset Pipeline?

Asset Bundle, welches man reinladen kann. Oder auch ein Unity Package.

Was ist die Umgebung, in der die Asset Pipeline betrieben wird?

Auf einem der gängigen Cloud Anbieter, einfach weil man das schnell skalieren kann. Sowas wie Azure, Amazon. Oder auch Google Cloud, wobei ich da nicht viel Erfahrung habe.

Wenn ich die Entscheidung habe, würde ich auf Azure oder Amazon setzen.

Wo siehst du alles Schnittstellen mit der bestehenden Systemlandschaft?

Die Schnittstellen wären eine Joberstellung. Dass ein Durchlauf erstellt werden kann. Und wenn das fertig ist, das Gegenstück davon, das fertige Asset in das Zielsystem überführen. Eventuell zusätzlich ein Status, sodass man weiß, wie der Fortschritt ist.

Folgefrage: Wo wäre diese Schnittstelle untergebracht? Wie sehe diese Schnittstelle aus?

Die Idee wäre, dass die Schnittstelle im Beispiel von Azure, die von der Pipeline ist. Die Pipeline kommuniziert mit der VM und baut. Dabei sollte es völlig egal sein, in was für einer VM gebaut wird. Der Request von Außerhalb hat nichts damit zu tun mit dem Bauen an sich. Es kann eine beliebige VM laufen, es sollte gekapselt sein. Es sollte also keine Kommunikation zwischen XR Cloud und der VM geben.

Welche Render Pipelines sollten alles unterstützt werden?

URP und Built-In sollten unterstützt werden. HDRP ist ein Bonus. Und Custom ist so eine Frage… Jemand der eine eigene Scripted Render Pipeline fährt ist nicht die Zielgruppe einer Asset Pipeline, weil da so viel anders passiert. Es ist da so customized, dass es ist schwer, da ein generisches Produkt zu bauen.

Welche Build Platforms sollten unterstützt werden?

Das wichtigste sind die drei Hauptplattformen, Windows, Android und iOS. Darüber hinaus ist natürlich Universal Windows Platform für Mixed Reality Anwendungsszenarien gut. Das sind die Wichtigsten meiner Meinung nach. Alle weiteren Plattformen sind eher spezifisch.

Welche Unity Versionen sollten unterstützt werden? Sollte man auch Alpha und Beta unterstützen?

Am Besten 2019 und 2020. Das ist eigentlich die Mehrheit der aktuellen Projekte. 2021 ist ja je nach Version noch Alpha und Beta. Da sehe ich jetzt nicht, dass man das unbedingt unterstützen muss.

Wie werden Kunden später die Asset Pipeline nutzen?

Das kommt aufs Vertriebsmodell an. Am Besten wäre es natürlich integriert in die bestehende SAP Cloud-Landschaft. Wenn man bereits die Consumption Based Cloud Services nutzt, bei dem man sowieso pro Benutzung zahlt. Ist es in dieses Modell integriert wird pro „Bauen“ bezahlt, mit Cloud Credits.

Auf der anderen Seite gibt es auch den Subscription Based Ansatz. Man mietet eine Anzahl von VMs und baut mit diesen. Wobei ich Consumption Based besser finde, weil man dort die Pipelines und VMs mit anderen Kunden teilt. Wenn ein Kunde im Moment nichts baut, kann der andere Kunde das nutzen. Das ist dann super effizient und insgesamt kann man das für den Kunden dann billiger anbieten.

Abschluss

Ist aktuelle Pipeline das, was wir wollten? Was fehlt?

Ich kenne die Entwicklung, aber nicht im Detail. Ich kann davon reden, was ich gesehen habe. Ja, die meisten initialen Dinge die wir haben wollen sind erfüllt. Mit zusätzlichen Sachen wie, dass alle Unity Versionen automatisch nachinstallieren kann. Das geht glaube ich bereits übers initiale Ziel hinaus. Die erste Version hätte auch ein paar feste Unity Versionen unterstützen können.

Wichtig ist auch noch für die zukunft das UX handling in Verbindung mit der Pipeline. Wenn man auf die Knöpfe drückt, soll man als Nutzer ein gutes UX Feeling hat und man sieht, was gerade ausgeführt worden ist, dass sich der Status ändert, nachdem man irgendwo draufgedrückt hat. Da weiß ich nicht im Detail wie der Fortschritt ist. Paar Dinge waren bereits ziemlich gut, aber sicherlich kann man viel von der UX Schnittstelle zwischen Pipeline und XR Cloud machen.

A.0.2. Interview mit Experte B

Angaben zum Interview:

- **Interviewpartner**: Experte B
- **Rolle**: Unity Entwickler, Smart Asset Entwickler
- **Datum**: 21.06.2021, 11:30 – 12:00
- **Ort**: Microsoft Teams Meeting
- **Durchgeführt von**: Enes Ördek
- **Einverständniserklärung**: Ja
- **Typ**: mündlich
- **Transkripiert von**: Enes Ördek

Einstiegsfragen

Wie lange arbeitest du bereits mit Unity?

Es ist nun schon fast 5 Jahre her, seit ich mit Unity arbeite.

Was hast du bereits mit Smart Assets /Asset Bundles gearbeitet?

Verschiedenes. Wir haben die Smart Asset Technologie um 3D Assets um in Runtime reinzuladen. Ich habe beim LKA at Runtime Panoramen reingeladen, um Memory Management zu handeln.

Ich entwickle auch das Smart Asset Plugin weiter und treibe es vorran durch die Entwicklung von Smart Labels und Smart Scanner.

Schlüsselfragen

Was stellst du dir unter einer Asset Pipeline allgemein vor?

Es gibt viele Definitionen von einer Asset Pipeline, und der Begriff ist breit gefasst. Was wir darunter verstehen ist etwas automatisiertes. Also etwas, was den manuellen Aufwand reduziert und eine Aufgabe skalierbar macht. Es gibt Semi-Automatisiertes, es gibt Voll-Automatisiertes. Nicht alles lässt sich immer automatisieren. Was ich unter Pipeline verstehe ist, dass man Aufgaben an Maschinen abgeben kann, die in eine Queue gesetzt werden, und die Sachen zurückgeben. Meistens handelt es sich um lange Aufgaben wie das Bauen von Asset Bundles, die man in einer Cloud Lösung skalieren kann. Eine Asset Pipeline ist customizeable und soll die Arbeitszeit des Mitarbeiters reduzieren.

Welche Probleme gibt es aktuell und welche Probleme soll die Asset Pipeline lösen?

Aktuell ist das Problem, dass wir immernoch manuell die Smart Assets bauen müssen, und je nach der Größe des Modells kann es viel Zeit in Anspruch nehmen, die ganzen Binaries zu bauen. Das ist das eine und ein großer Faktor. Das blockiert den Entwicklerrechner. Man kann das in einer Azure Devops Umgebung gut anwenden und skalieren.

Der andere Faktor ist langfristig: Wir haben die Anbindung an das SAP EPD System, in dem die ganzen 3D Modelle des Kunden sind. Wir wollen die Möglichkeit haben, dass der Kunde mit einem Klick seine 3D Modelle in ein Smart Asset umwandeln kann, ohne Sachen selber bauen zu müssen.

Welche Eingangsdaten hat die Asset Pipeline?

Das hängt stark von der Implementationsebene ab. Das Wichtige ist, EPD Modelle zu bauen, und da müssen wir wissen: Welche Modelle möchten wir bauen? Zu welchen Platformen brauchen wir diese Builds? Die meisten Kunden kennen sich mit Unity und 3D nicht gut aus, aber für die Fachkunden brauchen wir die Unity Version, die man definieren kann. Für noch fortgeschrittenere Kunden kann auch die Render Pipeline festgelegt werden.

Wo soll das rausgespeichert werden? Aber das ist klar, es soll in die XR Cloud gespeichert werden. Was ist der Smart Asset Name? Was ist die Priorität? Ist es high-priority? Ist es low-priority?

Man kann für die fortgeschritteneren Nutzer: Welche Shader können genutzt werden? Das hängt dann aber auch vom EPD Modell ab, das muss dann gemappt werden und ist eventuell too much. Ich glaube das ist das Gängigste, was man so man brauch.

Was ist die Ausgabe der Asset Pipeline? Welche Softwareartefakte gibt es?

Es hängt wieder von der Implementation ab. Aktuell bedienen wir nur Unity, da brauchen wir die Unity Asset Bundles bzw Smart Assets. Falls wir irgendwann die Unreal Engine bedienen brauchen wir noch was anderes. Die Smart Asset Bundles bundlen alles eigentlich rein, also sowas wie Textures usw.. Asset Bundles sind für Runtime gedacht, und für die Editor Time brauchen wir auch noch etwas. Bei uns gibt es da noch die Unity Packages. Dann ist da die Frage, ob es sinnvoll ist, für die Zukunft auch andere Datenformate bereitstellt. Sowas wie .fbx .obj usw. Wenn ein Kunde möchte nicht nur die vds Datei vom Modell, sondern eine normale obj mit Textures. Das könnte man zusätzlich noch haben.

Was ist die Umgebung, in der die Asset Pipeline betrieben wird?

Es sind zwei Komponenten: Einmal die Steuerungskomponente, die diese Pipeline steuert und einmal die einzelnen Workers, die VMs oder Maschinen, die arbeiten. Dann gibt es noch die Frage, worüber man das Ganze triggered. Wird das über die Cloud, über einen Desktop Client? Wo wird das gespeichert? In der Cloud? Im NAS? Wo liegen die Workers? Sind es Maschinen, die im internen Netz liegen? Ist es eine Cloud Lösung?

Wo siehst du alles Schnittstellen mit der bestehenden Systemlandschaft?

Wir haben natürlich die XR Cloud Schnittstelle. Das ist ein großer Faktor. Wir haben die Azure Devops Schnittstelle. Das ist aktuell das Skalierungstool und Pipelinetool der Wahl bei uns. Wir haben die Schnittstelle mit dem Smart Asset Plugin und dem Bundler System. Und wir haben das EPD Plugin als Schnittstelle.

Dann ist die Frage, wo triggert man den Build Prozess. Das ist dann sehr wahrscheinlich in der Cloud. Oder man hat da zusätzlich noch ein Desktop Client.

Wie kann die Pipeline konfiguriert werden?

Das ist ähnlich wie mit den Input Parametern. Was ist da noch zusätzlich zu sagen? Die Konfiguration kann die Möglichkeit geben, dass man zusätzlich noch Ausgabemöglichkeiten hat. Nicht nur Unity Asset Bundles, sondern weitere, falls weitere Platformen dazu kommen.

Welche Render Pipelines sollten alles unterstützt werden?

Die meisten Kunden von uns werden die Built-In Pipeline nutzen und das wird weiterhin so bleiben. Man kann URP und HDRP unterstützen. Aber das geht auch nur bis zu einem gewissen Grad, weil die Shader entsprechend gemapped werden müssen. Diese können aber nicht einfach automatisch umgewandelt werden. Also müsste hier die Verantwortung beim Kunden liegen. Wenn er was einricht, dann sollte das für die Built-In Render Pipeline sein.

Wenn ein Kunde etwas für URP haben möchte, sollte der Kunde das selbständig abbilden/umstellen und dann kann er für URP bauen. Ich würde nicht hergehen und sagen, dass er sein Smart Asset Bundle für alle drei Pipelines direkt bauen kann.

Man kann aber sagen, dass man eine bestimmte Grundfunktionalität bei HDRP und URP unterstützen kann. Ich würde die Verantwortung aber eher beim Kunden sehen.

Welche Build Platforms sollten unterstützt werden? (was ist mit Lumin?)

Ich hätte alle Unity Platformen, die der Smart Asset Bundler unterstützt, ebenfalls unterstützen. Die Wichtigsten natürlich: Windows, Mac, Linux hätte ich vielleicht gelassen, Android, iOS, UWP. Den Rest hätte ich sein gelassen.

Manches wird ja nicht supported auf allen Platformen. Darauf haben wir kein Einfluss. Das würde da dann in der Verantwortung des Nutzers liegen, der für eine bestimmte Platform bauen möchte.

Welche Unity Versionen sollten unterstützt werden?

Das hängt wieder vom Kunden ab. LTS ist immer gut. Stabile Versionen sind immer gut. Man kann 2019 und 2020 den LTS Stream unterstützen. Ob man die aller neuesten, den 2021 Stream mit einbindet, kann man eigentlich machen. Da die APIs sich nicht groß ändern für das Builden, ist es nicht so schlimm. Alpha und Beta würde ich nur machen, wenn es gewünscht ist.

Wie werden Kunden später die Asset Pipeline nutzen?

Die antizipierte Asset Pipeline wird so laufen, dass in ganz ferner Zukunft, ein EPD Kunde sein Modell hat, und im EPD Viewer auf „Bauen für Unity“ klickt. Mittelfristig wird es aus der XR Cloud rausgetiggert.

Abschluss

Ist aktuelle Pipeline das, was wir wollten? Wie ist dein Eindruck?

Es sieht gut aus. Das Ziel war ja, EPD Modelle mit einem Knopfdruck (bzw mit wenigen Knopfdrucken) ohne Unity zu öffnen, als Smart Asset bauen zu können. Und das kann sie.

Über das „Wo“ das getriggert wird, ist glaube ich erstmal relativ egal. Das kann man immernoch auf EPD Seite steuern. Aktuell wird es auf der XR Cloud gesteuert. Das kann man langfristig umbauen. An sich hat sie alles. Wir haben die Geschichte mit den Shadern und den Render Pipelines. Aber da weiß ich nicht ganz genau, wo der Stand da ist. Für das, was wir bisher an Kunden haben, reicht das.

A.0.3. Interview mit Experte C

Angaben zum Interview:

- **Interviewpartner**: Experte B
- **Rolle**: Entwickler, Architekt
- **Datum**: 21.06.2021, 11:00 – 11:30
- **Ort**: Microsoft Teams Meeting
- **Durchgeführt von**: Enes Ördek
- **Einverständniserklärung**: Ja
- **Typ**: mündlich
- **Transkripiert von**: Enes Ördek

Einstiegsfragen

Wie lange arbeitest du bereits mit Unity?

Mehrere Jahre schon, beginnend mit Oculus 2. Das war vermutlich so 2014. Ich war damals im Banking Umfeld, im Innovationslab und hatte dort angefangen zu experimentieren. Die Technik hatte erst mit DirectX programmiert. Damit konnte man mit Blöcken was bauen, die meisten Objekte in der Welt bestehen aus Blöcken. Aber das war mühsam und dann bin ich auf Unity aufmerksam geworden. Ich habe im Banking Insurance Umfeld die eine oder andere App damit gebaut.

Was hast du bereits mit Smart Assets /Asset Bundles gearbeitet?

Ja. In eurem Umfeld, oder auch im Retail Szenario, um Dinge abzuspeichern und eine Quelle zu haben, um komprimiert Assets dynamisch verfügbar zu haben und nicht nur hardcoded auszuliefern.

Schlüsselfragen

Was stellst du dir unter einer Asset Pipeline allgemein vor?

Im Grunde genommen eine Möglichkeit automatisiert Assets verwenden zu können in der eigenen Anwendung.

Welche Probleme gibt es aktuell und welche Probleme soll die Asset Pipeline lösen?

Hauptproblem sehe ich bei der manuellen Nacharbeit. Aber ich bezweifle, dass eine Asset Pipeline das Problem lösen kann. Die Sache ist die, wir haben hochauflösende CAD Zeichnungen. Je nach Anwendung brauchen wir nicht alle Details, nicht die vollständige Größe der Dateien. Das automatisierte Komprimieren ist sehr schwierig, es gibt Lösungen im Markt wie Pixyz, die das relativ gut können, aber mein Gefühl bleibt, dass du immer einen manuellen Designer brauchst, ein 3D Graphiker, ein UX Experte, der die Objekte vorbereitet. Momentan sehe ich da kein Weg vorbei. Zweite große Herausforderung: Die Asset Dimensionen. Jeder designed die

so, wie er möchte. Es gibt kein Standart, es wird kein Standart genutzt, der international sagt, wie groß so ein Objekt ist.

Der Unity Entwickler hat ein Flugzeug, und muss das ggf. skalieren, weil das nicht in Meter, Einheiten, Unity stattgefunden hat. Das sehe ich als zwei große Dinge, die es zu lösen gilt.

Welche Eingangsdaten hat die Asset Pipeline? EPD Parameter? Package? Konfiguration?

Es gibt sehr viele verschiedene Formate. Mit Visual Enterprise haben wir die Möglichkeit, verschiedene Formate zu konvertieren. Eine Pipeline sollte in der Lage sein, die gängigen Formate zu beherrschen, oder ggf. im PReprocessing zu den gängigen Formaten zu konvertieren. Im Unity Umfeld würde ich FBX und OBJ als standart sehen.

Was ist die Ausgabe der Asset Pipeline? Was wird als Build Artefakt erzeugt?

Entweder Unity Prefabs oder Unity FBX Objekte.

Was ist die Umgebung, in der die Asset Pipeline betrieben wird?

Cloud Umgebungen.

Wo siehst du alles Schnittstellen mit der bestehenden Systemlandschaft?

Überall zu Anwendungen wo 3D Objekte verwendet werden.

Welche Render Pipelines sollten alles unterstützt werden?

Keine spezielle Meinung.

Welche Build Platforms sollten unterstützt werden?

Alle gängigen Desktop Umgebungen wie MacOS, Windows. Dann die gängisten AR Plattformen, also Hololens. Die gängisten VR Plattformen. Die vollständige Unity Palette, außer den Exoten wie TVOS oder Nintendo.

Welche Unity Versionen sollten unterstützt werden?

Die jeweils neuesten Versionen. Also von 2019, 2020. Aber auch die neueste Beta.

Welche Use Cases sollte die Asset Pipeline so typischerweise abdecken?

Hatte ich anfangs schon genannt, automatisiertes Bereitstellen ohne großen manuellen Eingriff.

Wie werden Kunden später die Asset Pipeline nutzen? Werden sie überhaupt die Asset Pipeline nutzen?

Wenn die Apps eigene Apps bauen, dann sicherlich. Die Pipeline dient eher dem Entwickeln der Apps, statt dem Konsumieren.

Abschluss

Abschlussfragen wurden übersprungen, weil Experte nicht bei der Vorstellung der Asset Pipeline anwesend war.

A.0.4. Unity Hub Help

```
PS C:\Users> Start-Process -Wait "C:\Program Files\Unity Hub\Unity Hub.exe" -
    ArgumentList "-- --headless help"
Commands:
       editors
              description: list the releases and installed editors
              alias: e
              example: Unity\ Hub.app/Contents/MacOS/Unity\ Hub -- --headless
                  editors -r
              options:
                     [default]            list of available releases and
                         installed editors on your machine combined
                     --releases|-r  only list of available releases promoted by
                         Unity
                     --installed|-i only list of installed editors on your machine

       install-path
              description: set/get the path where the Unity editors will be
                  installed
              alias: ip
              example: Unity\ Hub.app/Contents/MacOS/Unity\ Hub -- --headless ip -
                  s /Applications/Unity/Hub/Editor/
              options:
                     --set|-s <path>      set the install path to the given path
                     --get|-g                    returns the install path

       install
              description: installs a new editor either from the releases list or
                  archive
              alias: i
              example: Unity\ Hub.app/Contents/MacOS/Unity\ Hub -- --headless
                  install --version 2019.1.11f1 --changeset 9b001d489a54
              options:
                     --version|-v <version>       editor version to be installed (
                         e.g. 2019.1.11f1) - required
                     --changeset|-c <changeset> changeset of the editor if it is
                         not in the release list (e.g. 9b001d489a54)
                                                                  - required
                                                                      if the
                                                                      version
                                                                       is not
                                                                       in the

                                                                      releases
                                                                      . see
                                                                      editors
                                                                       -r
                     --module|-m <moduleid>       see install-modules for more
                         information

       install-modules
              description: download and install a module (e.g. build support) to
                  an installed editor
```

```
alias: im
       example: Unity\ Hub.app/Contents/MacOS/Unity\ Hub -- --
           headless install-modules --version 2019.1.11f1 -m ios -m
           android
options:
       --version|-v <version> version of the editor to add the
           module to - required
       --module|-m <moduleid> the module id. The followings are the
           available values depending on version. You can specify
           multiple values.
                                                    Documentation:
                                                        documentation
                                                    Standard Assets:
                                                        standardassets
                                                    Example Project:
                                                        example
                                                    Android Build
                                                        Support:
                                                        android
                                                    iOS Build Support:
                                                         ios
                                                    tvOS Build Support
                                                        : appletv
                                                    Linux Build
                                                        Support: linux
                                                        -mono
                                                    SamsungTV Build
                                                        Support:
                                                        samsung
                                                    Tizen Build
                                                        Support: tizen
                                                    WebGL Build
                                                        Support: webgl
                                                    Windows Build
                                                        Support:
                                                        windows
                                                    Facebook Gameroom
                                                        Build Support:
                                                         facebook-
                                                        games
                                                    MonoDevelop /
                                                        Unity Debugger
                                                        : monodevelop
                                                    Vuforia Augmented
                                                        Reality
                                                        Support:
                                                        vuforia-ar
                                                    Language packs:
                                                        language-ja,
                                                        language-ko,
                                                        language-zh-cn
                                                        , language-zh-
```

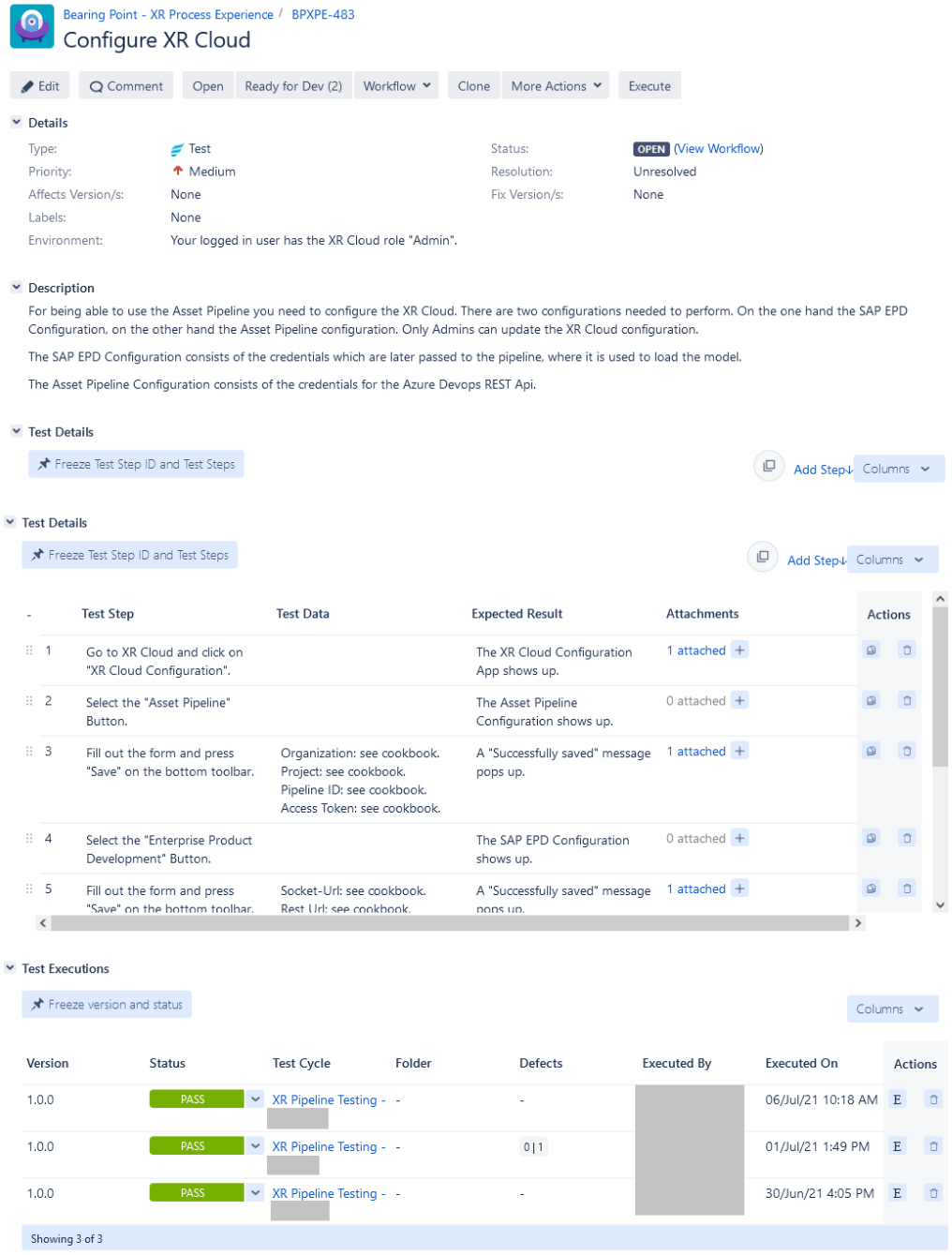
```
                                                    hant, language
                                                    -zh-hans
                                                Mac Build Support
                                                    (IL2CPP): mac-
                                                    il2cpp
                                                Windows Build
                                                    Support (Mono)
                                                    : windows-mono
                                                Android SDK & NDK
                                                    Tools: android
                                                    -sdk-ndk-tools
                                                OpenJDK: android-
                                                    open-jdk
                                                Lumin OS (Magic
                                                    Leap) Build
                                                    Support: lumin
  --childModules|--cm   automatically installs all child
      modules of selected modules


PS C:\Users>
```

Listing A.1: UnityHubHelp

Anhang B

Zweiter Anhang

B.1. Testergebnisse



Bearing Point - XR Process Experience / BPXPE-483

Configure XR Cloud

Edit | Comment | Open | Ready for Dev (2) | Workflow | Clone | More Actions | Execute

Details

Type:	Test	Status:	OPEN (View Workflow)
Priority:	Medium	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	None
Labels:	None		
Environment:	Your logged in user has the XR Cloud role "Admin".		

Description

For being able to use the Asset Pipeline you need to configure the XR Cloud. There are two configurations needed to perform. On the one hand the SAP EPD Configuration, on the other hand the Asset Pipeline configuration. Only Admins can update the XR Cloud configuration.

The SAP EPD Configuration consists of the credentials which are later passed to the pipeline, where it is used to load the model.

The Asset Pipeline Configuration consists of the credentials for the Azure Devops REST Api.

Test Details

Freeze Test Step ID and Test Steps | Add Step | Columns

Test Details

Freeze Test Step ID and Test Steps | Add Step | Columns

-	Test Step	Test Data	Expected Result	Attachments	Actions
1	Go to XR Cloud and click on "XR Cloud Configuration".		The XR Cloud Configuration App shows up.	1 attached	
2	Select the "Asset Pipeline" Button.		The Asset Pipeline Configuration shows up.	0 attached	
3	Fill out the form and press "Save" on the bottom toolbar.	Organization: see cookbook. Project: see cookbook. Pipeline ID: see cookbook. Access Token: see cookbook.	A "Successfully saved" message pops up.	1 attached	
4	Select the "Enterprise Product Development" Button.		The SAP EPD Configuration shows up.	0 attached	
5	Fill out the form and press "Save" on the bottom toolbar.	Socket-Url: see cookbook. Rest Url: see cookbook.	A "Successfully saved" message pops up.	1 attached	

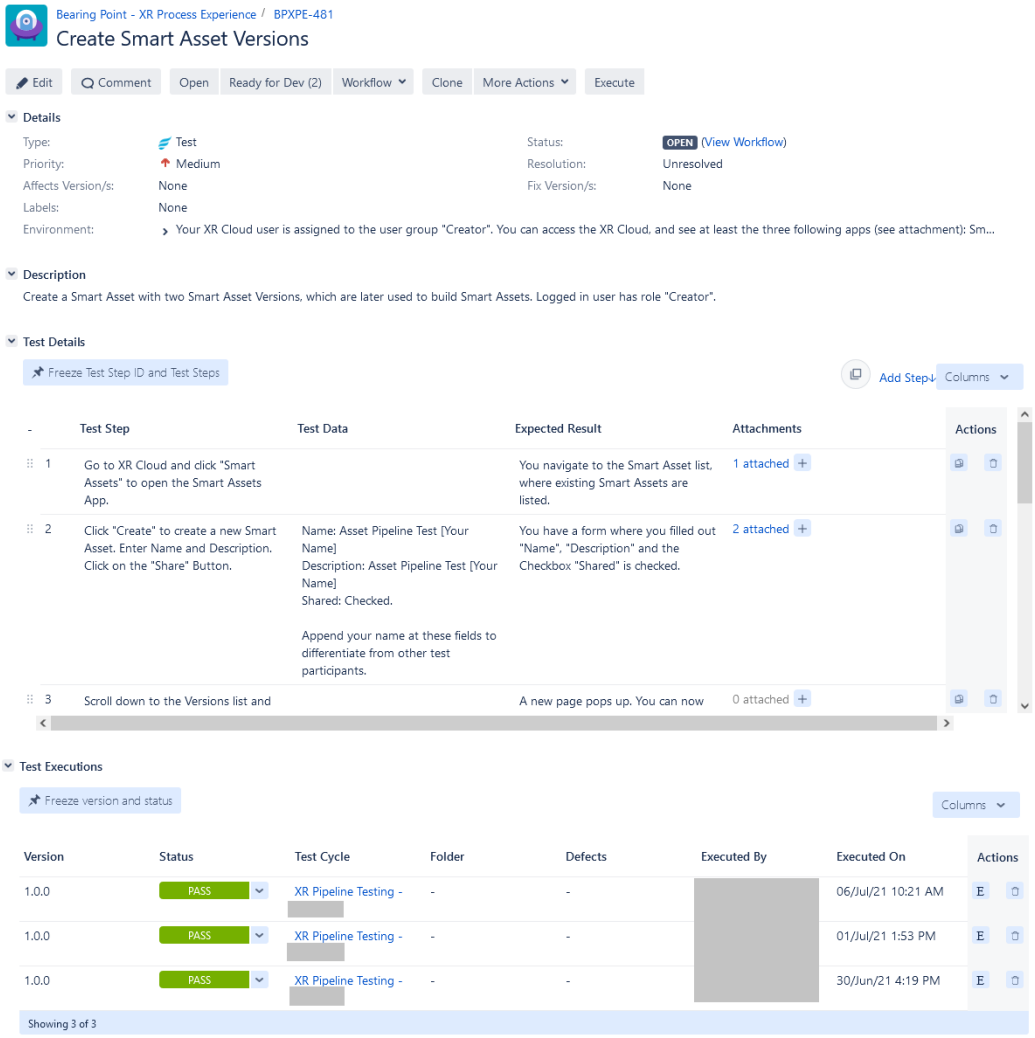
Test Executions

Freeze version and status | Columns

Version	Status	Test Cycle	Folder	Defects	Executed By	Executed On	Actions
1.0.0	PASS	XR Pipeline Testing -	-	-		06/Jul/21 10:18 AM	E
1.0.0	PASS	XR Pipeline Testing -	-	0 | 1		01/Jul/21 1:49 PM	E
1.0.0	PASS	XR Pipeline Testing -	-	-		30/Jun/21 4:05 PM	E

Showing 3 of 3

Abbildung B.1.: Ergebnisse des ersten Testfalls



Bearing Point - XR Process Experience / BPXPE-481

Create Smart Asset Versions

Edit | Comment | Open | Ready for Dev (2) | Workflow | Clone | More Actions | Execute

Details

Type:	Test	Status:	OPEN (View Workflow)
Priority:	Medium	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	None
Labels:	None		
Environment:	Your XR Cloud user is assigned to the user group "Creator". You can access the XR Cloud, and see at least the three following apps (see attachment): Sm...		

Description

Create a Smart Asset with two Smart Asset Versions, which are later used to build Smart Assets. Logged in user has role "Creator".

Test Details

Freeze Test Step ID and Test Steps | Add Step | Columns

-	Test Step	Test Data	Expected Result	Attachments	Actions
1	Go to XR Cloud and click "Smart Assets" to open the Smart Assets App.		You navigate to the Smart Asset list, where existing Smart Assets are listed.	1 attached	
2	Click "Create" to create a new Smart Asset. Enter Name and Description. Click on the "Share" Button.	Name: Asset Pipeline Test [Your Name] Description: Asset Pipeline Test [Your Name] Shared: Checked. Append your name at these fields to differentiate from other test participants.	You have a form where you filled out "Name", "Description" and the Checkbox "Shared" is checked.	2 attached	
3	Scroll down to the Versions list and		A new page pops up. You can now	0 attached	

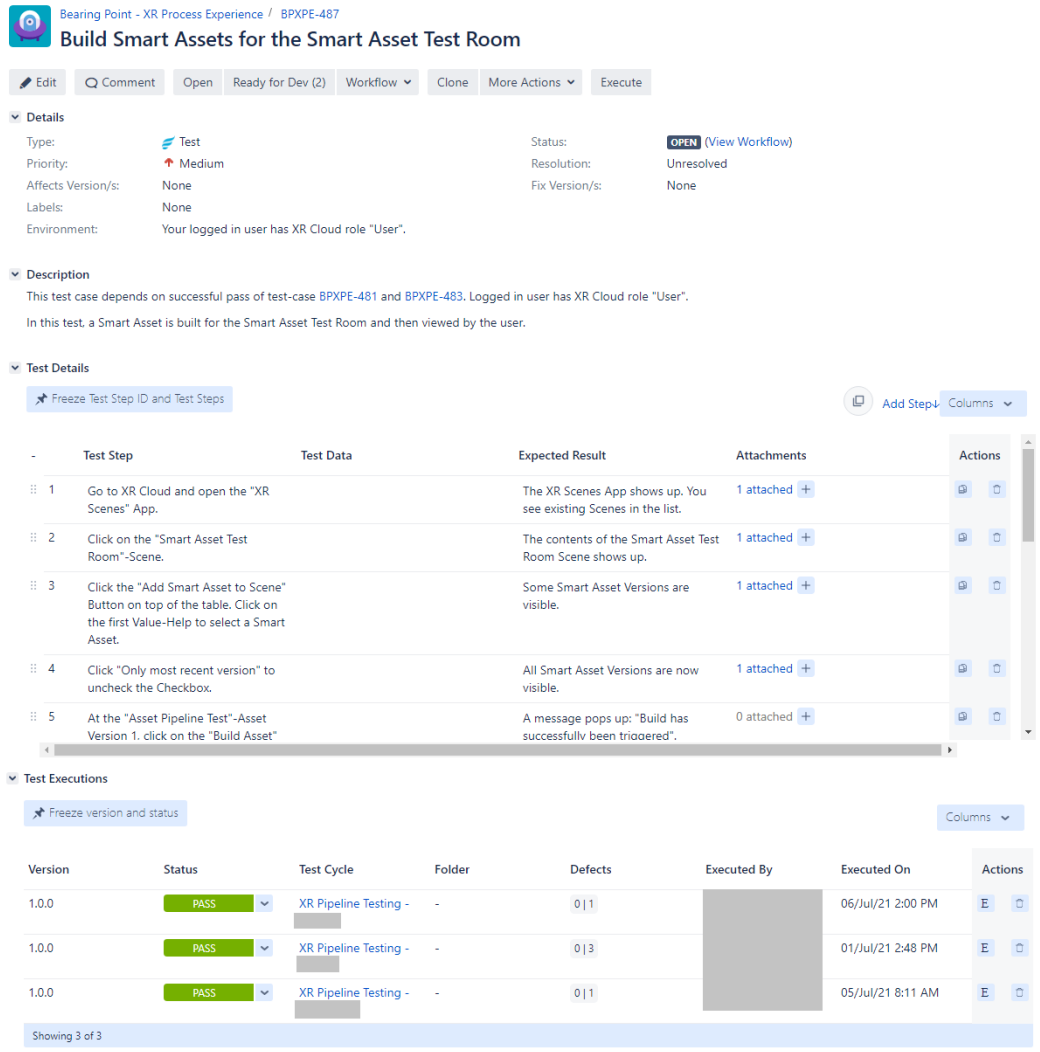
Test Executions

Freeze version and status | Columns

Version	Status	Test Cycle	Folder	Defects	Executed By	Executed On	Actions
1.0.0	PASS	XR Pipeline Testing -	-	-		06/Jul/21 10:21 AM	
1.0.0	PASS	XR Pipeline Testing -	-	-		01/Jul/21 1:53 PM	
1.0.0	PASS	XR Pipeline Testing -	-	-		30/Jun/21 4:19 PM	

Showing 3 of 3

Abbildung B.2.: Ergebnisse des zweiten Testfalls



Bearing Point - XR Process Experience / BPXPE-487

Build Smart Assets for the Smart Asset Test Room

Edit | Comment | Open | Ready for Dev (2) | Workflow | Clone | More Actions | Execute

Details

Type:	Test	Status:	OPEN (View Workflow)
Priority:	Medium	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	None
Labels:	None		
Environment:	Your logged in user has XR Cloud role "User".		

Description

This test case depends on successful pass of test-case BPXPE-481 and BPXPE-483. Logged in user has XR Cloud role "User".

In this test, a Smart Asset is built for the Smart Asset Test Room and then viewed by the user.

Test Details

Freeze Test Step ID and Test Steps | Add Step | Columns

-	Test Step	Test Data	Expected Result	Attachments	Actions
1	Go to XR Cloud and open the "XR Scenes" App.		The XR Scenes App shows up. You see existing Scenes in the list.	1 attached	
2	Click on the "Smart Asset Test Room"-Scene.		The contents of the Smart Asset Test Room Scene shows up.	1 attached	
3	Click the "Add Smart Asset to Scene" Button on top of the table. Click on the first Value-Help to select a Smart Asset.		Some Smart Asset Versions are visible.	1 attached	
4	Click "Only most recent version" to uncheck the Checkbox.		All Smart Asset Versions are now visible.	1 attached	
5	At the "Asset Pipeline Test"-Asset Version 1, click on the "Build Asset"		A message pops up: "Build has successfully been triggered".	0 attached	

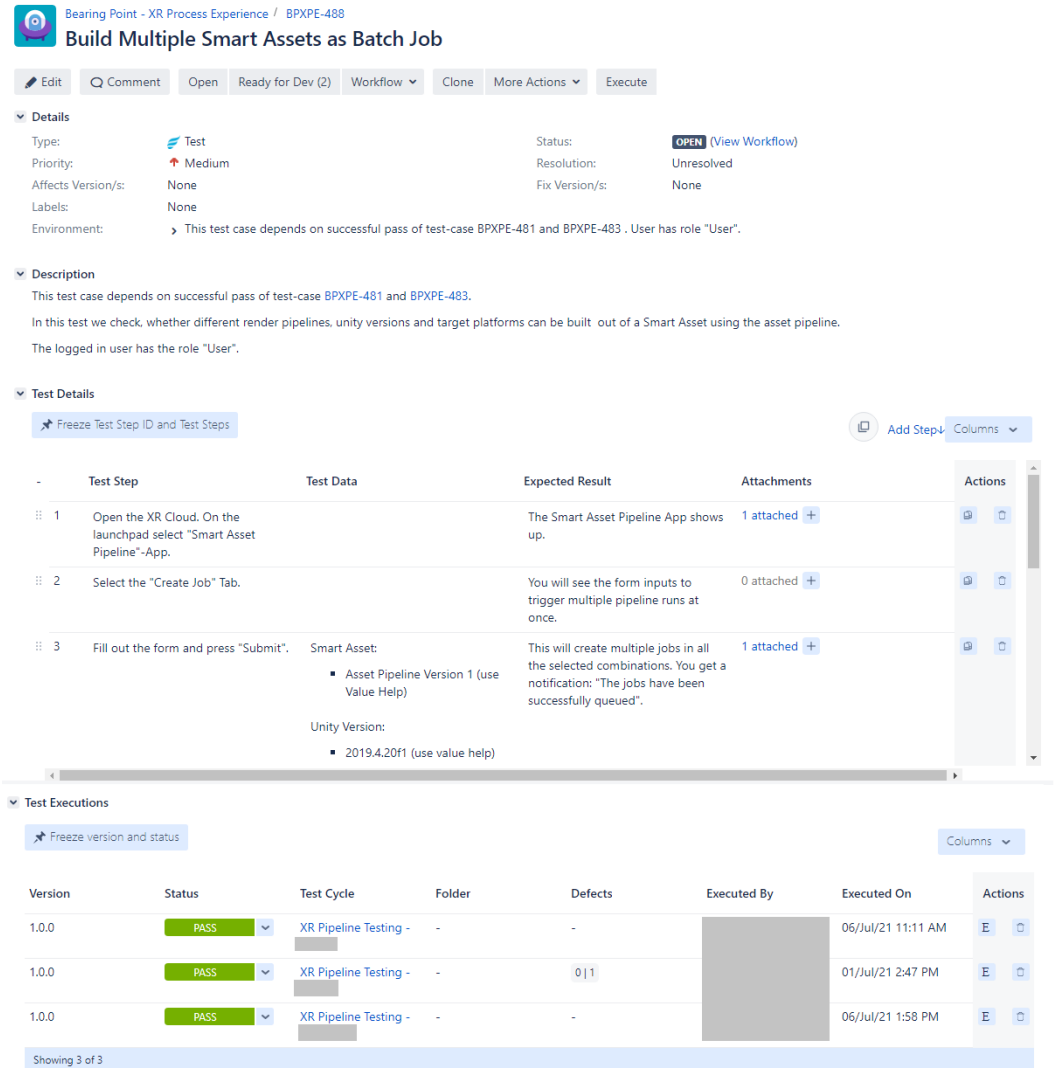
Test Executions

Freeze version and status | Columns

Version	Status	Test Cycle	Folder	Defects	Executed By	Executed On	Actions
1.0.0	PASS	XR Pipeline Testing -	-	0 | 1		06/Jul/21 2:00 PM	
1.0.0	PASS	XR Pipeline Testing -	-	0 | 3		01/Jul/21 2:48 PM	
1.0.0	PASS	XR Pipeline Testing -	-	0 | 1		05/Jul/21 8:11 AM	

Showing 3 of 3

Abbildung B.3.: Ergebnisse des dritten Testfalls



Bearing Point - XR Process Experience / BPXPE-488

Build Multiple Smart Assets as Batch Job

Edit | Comment | Open | Ready for Dev (2) | Workflow | Clone | More Actions | Execute

Details

Type:	Test	Status:	OPEN (View Workflow)
Priority:	Medium	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	None
Labels:	None		
Environment:	This test case depends on successful pass of test-case BPXPE-481 and BPXPE-483 . User has role "User".		

Description

This test case depends on successful pass of test-case BPXPE-481 and BPXPE-483.

In this test we check, whether different render pipelines, unity versions and target platforms can be built out of a Smart Asset using the asset pipeline.

The logged in user has the role "User".

Test Details

Freeze Test Step ID and Test Steps | Add Step | Columns

-	Test Step	Test Data	Expected Result	Attachments	Actions
1	Open the XR Cloud. On the launchpad select "Smart Asset Pipeline"-App.		The Smart Asset Pipeline App shows up.	1 attached	
2	Select the "Create Job" Tab.		You will see the form inputs to trigger multiple pipeline runs at once.	0 attached	
3	Fill out the form and press "Submit".	Smart Asset: ▪ Asset Pipeline Version 1 (use Value Help) Unity Version: ▪ 2019.4.20f1 (use value help)	This will create multiple jobs in all the selected combinations. You get a notification: "The jobs have been successfully queued".	1 attached	

Test Executions

Freeze version and status | Columns

Version	Status	Test Cycle	Folder	Defects	Executed By	Executed On	Actions
1.0.0	PASS	XR Pipeline Testing -	-	-		06/Jul/21 11:11 AM	E
1.0.0	PASS	XR Pipeline Testing -	-	0 | 1		01/Jul/21 2:47 PM	E
1.0.0	PASS	XR Pipeline Testing -	-	-		06/Jul/21 1:58 PM	E

Showing 3 of 3

Abbildung B.4.: Ergebnisse des vierten Testfalls

B.2. Datenreihen

B.2.1. Einfache Berechnung

Aufwand pro Smart Asset (Personenstunden)	0,083
Kosten Asset Pipeline pro Monat (Euro)	200
Euro pro Personenstunde (Euro)	100

Tabelle B.1.: Parameter einfache Berechnung

Anzahl Smart Assets	Kosten ohne Asset Pipeline	Preis pro Smart Asset Ohne Asset Pipeline	Kosten mit Asset Pipeline	Preis pro Smart Asset mit Asset Pipeline
1	8 €	8 €	200 €	200 €
2	17 €	8 €	200 €	100 €
3	25 €	8 €	200 €	67 €
4	33 €	8 €	200 €	50 €
5	42 €	8 €	200 €	40 €
6	50 €	8 €	200 €	33 €
7	58 €	8 €	200 €	29 €
8	66 €	8 €	200 €	25 €
9	75 €	8 €	200 €	22 €
10	83 €	8 €	200 €	20 €
11	91 €	8 €	200 €	18 €
12	100 €	8 €	200 €	17 €
13	108 €	8 €	200 €	15 €
14	116 €	8 €	200 €	14 €
15	125 €	8 €	200 €	13 €
16	133 €	8 €	200 €	13 €
17	141 €	8 €	200 €	12 €
18	149 €	8 €	200 €	11 €
19	158 €	8 €	200 €	11 €
20	166 €	8 €	200 €	10 €
21	174 €	8 €	200 €	10 €
22	183 €	8 €	200 €	9 €
23	191 €	8 €	200 €	9 €
24	199 €	8 €	200 €	8 €
25	208 €	8 €	200 €	8 €
26	216 €	8 €	200 €	8 €
27	224 €	8 €	200 €	7 €
28	232 €	8 €	200 €	7 €
29	241 €	8 €	200 €	7 €
30	249 €	8 €	200 €	7 €
31	257 €	8 €	200 €	6 €
32	266 €	8 €	200 €	6 €
33	274 €	8 €	200 €	6 €
34	282 €	8 €	200 €	6 €
35	291 €	8 €	200 €	6 €

Tabelle B.2.: Datenreihe einfache Berechnung Teil 1

Anzahl Smart Assets	Kosten ohne Asset Pipeline	Preis pro Smart Asset Ohne Asset Pipeline	Kosten mit Asset Pipeline	Preis pro Smart Asset mit Asset Pipeline
36	299 €	8 €	200 €	6 €
37	307 €	8 €	200 €	5 €
38	315 €	8 €	200 €	5 €
39	324 €	8 €	200 €	5 €
40	332 €	8 €	200 €	5 €
41	340 €	8 €	200 €	5 €
42	349 €	8 €	200 €	5 €
43	357 €	8 €	200 €	5 €
44	365 €	8 €	200 €	5 €
45	374 €	8 €	200 €	4 €
46	382 €	8 €	200 €	4 €
47	390 €	8 €	200 €	4 €
48	398 €	8 €	200 €	4 €
49	407 €	8 €	200 €	4 €
50	415 €	8 €	200 €	4 €
51	423 €	8 €	200 €	4 €
52	432 €	8 €	200 €	4 €
53	440 €	8 €	200 €	4 €
54	448 €	8 €	200 €	4 €
55	457 €	8 €	200 €	4 €
56	465 €	8 €	200 €	4 €
57	473 €	8 €	200 €	4 €
58	481 €	8 €	200 €	3 €
59	490 €	8 €	200 €	3 €
60	498 €	8 €	200 €	3 €
61	506 €	8 €	200 €	3 €
62	515 €	8 €	200 €	3 €
63	523 €	8 €	200 €	3 €
64	531 €	8 €	200 €	3 €
65	540 €	8 €	200 €	3 €
66	548 €	8 €	200 €	3 €
67	556 €	8 €	200 €	3 €
68	564 €	8 €	200 €	3 €
69	573 €	8 €	200 €	3 €
70	581 €	8 €	200 €	3 €

Tabelle B.3.: Datenreihe einfache Berechnung Teil 2

Anzahl Smart Assets	Kosten ohne Asset Pipeline	Preis pro Smart Asset Ohne Asset Pipeline	Kosten mit Asset Pipeline	Preis pro Smart Asset mit Asset Pipeline
71	589 €	8 €	200 €	3 €
72	598 €	8 €	200 €	3 €
73	606 €	8 €	200 €	3 €
74	614 €	8 €	200 €	3 €
75	623 €	8 €	200 €	3 €
76	631 €	8 €	200 €	3 €
77	639 €	8 €	200 €	3 €
78	647 €	8 €	200 €	3 €
79	656 €	8 €	200 €	3 €
80	664 €	8 €	200 €	3 €
81	672 €	8 €	200 €	2 €
82	681 €	8 €	200 €	2 €
83	689 €	8 €	200 €	2 €
84	697 €	8 €	200 €	2 €
85	706 €	8 €	200 €	2 €
86	714 €	8 €	200 €	2 €
87	722 €	8 €	200 €	2 €
88	730 €	8 €	200 €	2 €
89	739 €	8 €	200 €	2 €
90	747 €	8 €	200 €	2 €
91	755 €	8 €	200 €	2 €
92	764 €	8 €	200 €	2 €
93	772 €	8 €	200 €	2 €
94	780 €	8 €	200 €	2 €
95	789 €	8 €	200 €	2 €
96	797 €	8 €	200 €	2 €
97	805 €	8 €	200 €	2 €
98	813 €	8 €	200 €	2 €
99	822 €	8 €	200 €	2 €
100	830 €	8 €	200 €	2 €

Tabelle B.4.: Datenreihe einfache Berechnung Teil 3

B.2.2. Realistische Berechnung

Aufwand pro Smart Asset (Personenstunden)	4,73
Kosten Asset Pipeline pro Monat (Euro)	200
Euro pro Personenstunde (Euro)	100

Tabelle B.5.: Parameter realistische Berechnung

Anzahl Smart Assets	Kosten ohne Asset Pipeline	Preis pro Smart Asset Ohne Asset Pipeline	Kosten mit Asset Pipeline	Preis pro Smart Asset mit Asset Pipeline
1	473 €	473 €	200 €	200 €
2	946 €	473 €	200 €	100 €
3	1.419 €	473 €	200 €	67 €
4	1.892 €	473 €	200 €	50 €
5	2.365 €	473 €	200 €	40 €
6	2.838 €	473 €	200 €	33 €
7	3.311 €	473 €	200 €	29 €
8	3.784 €	473 €	200 €	25 €
9	4.257 €	473 €	200 €	22 €
10	4.730 €	473 €	200 €	20 €
11	5.203 €	473 €	200 €	18 €
12	5.676 €	473 €	200 €	17 €
13	6.149 €	473 €	200 €	15 €
14	6.622 €	473 €	200 €	14 €
15	7.095 €	473 €	200 €	13 €
16	7.568 €	473 €	200 €	13 €
17	8.041 €	473 €	200 €	12 €
18	8.514 €	473 €	200 €	11 €
19	8.987 €	473 €	200 €	11 €
20	9.460 €	473 €	200 €	10 €
21	9.933 €	473 €	200 €	10 €
22	10.406 €	473 €	200 €	9 €
23	10.879 €	473 €	200 €	9 €
24	11.352 €	473 €	200 €	8 €
25	11.825 €	473 €	200 €	8 €
26	12.298 €	473 €	200 €	8 €
27	12.771 €	473 €	200 €	7 €
28	13.244 €	473 €	200 €	7 €
29	13.717 €	473 €	200 €	7 €
30	14.190 €	473 €	200 €	7 €
31	14.663 €	473 €	200 €	6 €
32	15.136 €	473 €	200 €	6 €
33	15.609 €	473 €	200 €	6 €
34	16.082 €	473 €	200 €	6 €
35	16.555 €	473 €	200 €	6 €

Tabelle B.6.: Datenreihe realistische Berechnung Teil 1

Anzahl Smart Assets	Kosten ohne Asset Pipeline	Preis pro Smart Asset Ohne Asset Pipeline	Kosten mit Asset Pipeline	Preis pro Smart Asset mit Asset Pipeline
36	17.028 €	473 €	200 €	6 €
37	17.501 €	473 €	200 €	5 €
38	17.974 €	473 €	200 €	5 €
39	18.447 €	473 €	200 €	5 €
40	18.920 €	473 €	200 €	5 €
41	19.393 €	473 €	200 €	5 €
42	19.866 €	473 €	200 €	5 €
43	20.339 €	473 €	200 €	5 €
44	20.812 €	473 €	200 €	5 €
45	21.285 €	473 €	200 €	4 €
46	21.758 €	473 €	200 €	4 €
47	22.231 €	473 €	200 €	4 €
48	22.704 €	473 €	200 €	4 €
49	23.177 €	473 €	200 €	4 €
50	23.650 €	473 €	200 €	4 €
51	24.123 €	473 €	200 €	4 €
52	24.596 €	473 €	200 €	4 €
53	25.069 €	473 €	200 €	4 €
54	25.542 €	473 €	200 €	4 €
55	26.015 €	473 €	200 €	4 €
56	26.488 €	473 €	200 €	4 €
57	26.961 €	473 €	200 €	4 €
58	27.434 €	473 €	200 €	3 €
59	27.907 €	473 €	200 €	3 €
60	28.380 €	473 €	200 €	3 €
61	28.853 €	473 €	200 €	3 €
62	29.326 €	473 €	200 €	3 €
63	29.799 €	473 €	200 €	3 €
64	30.272 €	473 €	200 €	3 €
65	30.745 €	473 €	200 €	3 €
66	31.218 €	473 €	200 €	3 €
67	31.691 €	473 €	200 €	3 €
68	32.164 €	473 €	200 €	3 €
69	32.637 €	473 €	200 €	3 €
70	33.110 €	473 €	200 €	3 €

Tabelle B.7.: Datenreihe realistische Berechnung Teil 2

Anzahl Smart Assets	Kosten ohne Asset Pipeline	Preis pro Smart Asset Ohne Asset Pipeline	Kosten mit Asset Pipeline	Preis pro Smart Asset mit Asset Pipeline
71	33.583 €	473 €	200 €	3 €
72	34.056 €	473 €	200 €	3 €
73	34.529 €	473 €	200 €	3 €
74	35.002 €	473 €	200 €	3 €
75	35.475 €	473 €	200 €	3 €
76	35.948 €	473 €	200 €	3 €
77	36.421 €	473 €	200 €	3 €
78	36.894 €	473 €	200 €	3 €
79	37.367 €	473 €	200 €	3 €
80	37.840 €	473 €	200 €	3 €
81	38.313 €	473 €	200 €	2 €
82	38.786 €	473 €	200 €	2 €
83	39.259 €	473 €	200 €	2 €
84	39.732 €	473 €	200 €	2 €
85	40.205 €	473 €	200 €	2 €
86	40.678 €	473 €	200 €	2 €
87	41.151 €	473 €	200 €	2 €
88	41.624 €	473 €	200 €	2 €
89	42.097 €	473 €	200 €	2 €
90	42.570 €	473 €	200 €	2 €
91	43.043 €	473 €	200 €	2 €
92	43.516 €	473 €	200 €	2 €
93	43.989 €	473 €	200 €	2 €
94	44.462 €	473 €	200 €	2 €
95	44.935 €	473 €	200 €	2 €
96	45.408 €	473 €	200 €	2 €
97	45.881 €	473 €	200 €	2 €
98	46.354 €	473 €	200 €	2 €
99	46.827 €	473 €	200 €	2 €
100	47.300 €	473 €	200 €	2 €

Tabelle B.8.: Datenreihe realistische Berechnung Teil 3