LEMONTREE AUTOMATION

Wie Sie mit LemonTree Components in einer Build-Pipeline mittels Versionskontrolle **erfolgreich** arbeiten

Whitepaper

LieberLieber Software

INHALT

EINLEITUNG

Lifecycle Management ist ein wichtiges Thema bei der Entwicklung von cyber-physischen Systemen. Diese Systeme werden mit Hilfe von modelbasierten Ansätze spezifiziert, entwickelt und verifiziert. Dabei wird oft ein Gesamtmodell erstellt, das mehrere Teilsysteme oder Komponenten beinhaltet. Wird das Gesamtmodell unter Versionskontrolle gestellt, müssen alle Komponenten über ihren Lebenszyklus gemeinsam verwaltet werden (Releases, Varianten etc.).

Um für Teile des Modells bzw. einzelne Komponenten ein Lifecycle Management zu betreiben, muss das Modell aufgetrennt werden. Anschließend lassen sich die so entstandenen Teilmodelle einzeln pflegen. LemonTree Components ermöglicht es, Teile eines Enterprise Architect Modells als wiederverwendbare Komponenten zu spezifizieren und aus dem Modell herauszulösen. Diese Komponenten lassen sich anschließend als eigenständiges Modell bearbeiten, versionieren und ins Gesamtmodell zurückspielen.

Das Gesamtmodell wird abhängig von der Arbeitsweise regelmäßig aktualisiert und eine neue Revision der Komponenten eingespielt. Diese Aufgabe übernimmt LemonTree Automation, integriert in einer Build-Server Pipeline. So werden Modelle in gewohnte Prozesse integriert und Änderungen bleiben sicht- und prüfbar.

Im vorliegenden Whitepaper wird beschrieben, wie LemonTree Components und LemonTree Automation in einen DevOps Prozess integriert werden, um den Lebenszyklus von Teilmodellen effizient zu verwalten.

LEMONTREE **COMPONENTS**

LemonTree Components erlaubt die Aufteilung eines mit Enterprise Architect erstellten Modells in verschiedene Teilmodelle bzw. Komponenten. Diese lassen sich leicht verteilen, wiederverwenden und bei Bedarf auch wieder in das Gesamtmodell integrieren.

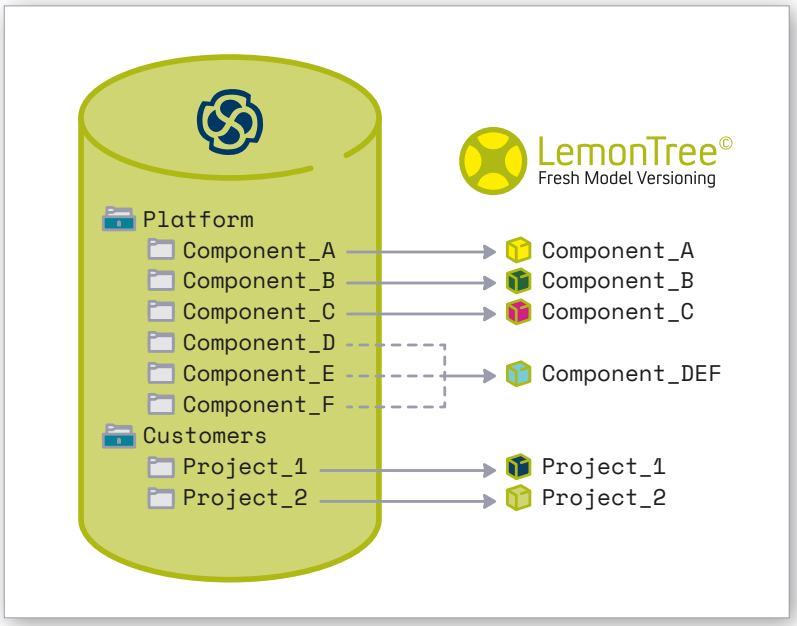


Abbildung 1: Vom Gesamt Modell zu Modellkomponenten

LemonTree Components werden mittels Export aus dem Modell extrahiert und mittels Import in ein anderes Modell importiert. Damit wird die verteilte Bearbeitung ganz gezielt auf bestimmte Teile des Modells eingeschränkt. Mit LemonTree lassen sich Unterschiede bei den bearbeiteten Komponenten erkennen und visualisieren. Nach der Anpassung der Komponente in einem sogenannten Arbeitsmodell lassen sich die Änderungen in das ursprüngliche Gesamtmodell zurückspielen. Wurde dabei der Inhalt der Komponenten auf beiden Seiten verändert (sowohl im Gesamt- als auch im Arbeitsmodell), ermöglicht LemonTree die Erstellung einer neuen, zusammengeführten Version der Komponente. Bei Vorliegen eines Konflikts zwischen den Änderungen unterstützt LemonTree bei der Auflösung und Konsolidierung.

LEMONTREE **AUTOMATION**

Die oben erwähnte Funktion von LemonTree Components ist sowohl mit dem LemonTree Addin als auch über das Kommandozeilen-Interface von LemonTree Automation ausführbar.

LemonTree unterstützt folgende Features:

- Export von LemonTree Components
- Import von LemonTree Components
- Merge von Enterprise Architect Modellen (nur ohne Konflikte; im Fall eines Konflikts bricht LemonTree mit einem ExitCode ab)
- Vergleich von Enterprise Architect Modellen
- Erzeugen von „Single File Sessions“ (Datei, welche die Diff-Information beinhaltet und die verwendete LemonTree Session reproduzieren kann)

LemonTree Automation wird primär im Kontext einer Build-Server Pipeline eingesetzt, um dort LemonTree Components zu aktualisieren (basierend auf Commits) oder Diff Sessions zu erzeugen, die als Review-Artefakt benutzt werden können.

Die nächsten Kapitel beschreiben einen möglichen Workflow, um LemonTree Components gemeinsam mit LemonTree Automation in einer Build-Server Pipeline zu integrieren.dierung.

BEISPIEL-WORKFLOW: INTEGRATION IN EINE BUILD-PIPELINE

Der folgende Workflow zeigt eine Möglichkeit, um mit LemonTree Components und Automation im Kontext einer Build-Pipeline zu arbeiten. Durch die Ausschöpfung aller Möglichkeiten der Tools entsteht dabei eine umfangreiche Toolkette. Ein Überblick über die unterschiedlichen Ausbaustufen wird im Kapitel „Verschiedene Ausbaustufen des Beispiel-Workflows“ gegeben.

Grundsätzlich lassen sich Enterprise Architect Modelle in Versionskontrollsystemen verwalten und somit in einer Build-Pipeline verwenden. Ein Problem dabei ist jedoch, dass unter Umständen ein monolithisches Gesamtmodell versioniert wird, das aus verschiedenen, eigenständigen Komponenten besteht. Zur Lösung dieses Problems wird das Modell in LemonTree Components zerteilt und jede Komponente in einem eigenen Versionskontroll-Repository (in diesem Beispiel Git Repository genannt) verwaltet.

Der erste Schritt in diesem Prozess ist die Spezifikation von LemonTree Components. Dabei wird definiert, welche Enterprise Architect Packages zu einer logischen Komponente zusammengefasst werden.

SPEZIFIKATION VON LEMONTREE COMPONENTS

Die Spezifikation einer LemonTree Component wird über das LemonTree Addin erstellt. Dabei werden die Pakete ausgewählt, die Teil der LemonTree Component werden sollen. Eine Komponente kann optional als „Read-Only“ markiert werden, wodurch sie nach dem Import in ein Arbeitsmodell nicht bearbeitbar ist. So ist die Komponente zwar verwendbar, lässt sich aber nicht modifizieren.

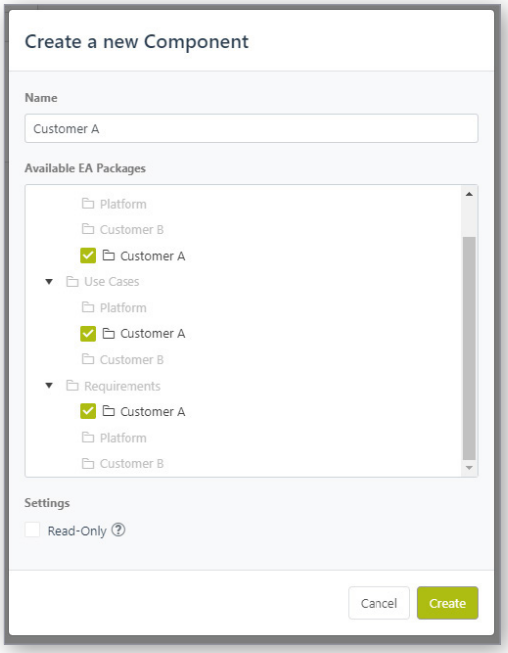


Abbildung 2: Spezifikation einer LemonTree Component, bestehend aus mehreren EA Paketen

Beim Export einer Komponente aus dem Modell sind deren Abhängigkeiten zu berücksichtigen. Diese Referenzen werden von LemonTree erkannt und dargestellt. Wichtig dabei sind die ausgehenden Referenzen, die eine Abhängigkeit darstellen.

ABHÄNGIGKEITEN VON LEMONTREE COMPONENTS

Abhängigkeiten zwischen Komponenten ergeben sich durch Referenzen im Modell, wie z.B. Classifier Referenzen, Relationen oder grafische Darstellungen.

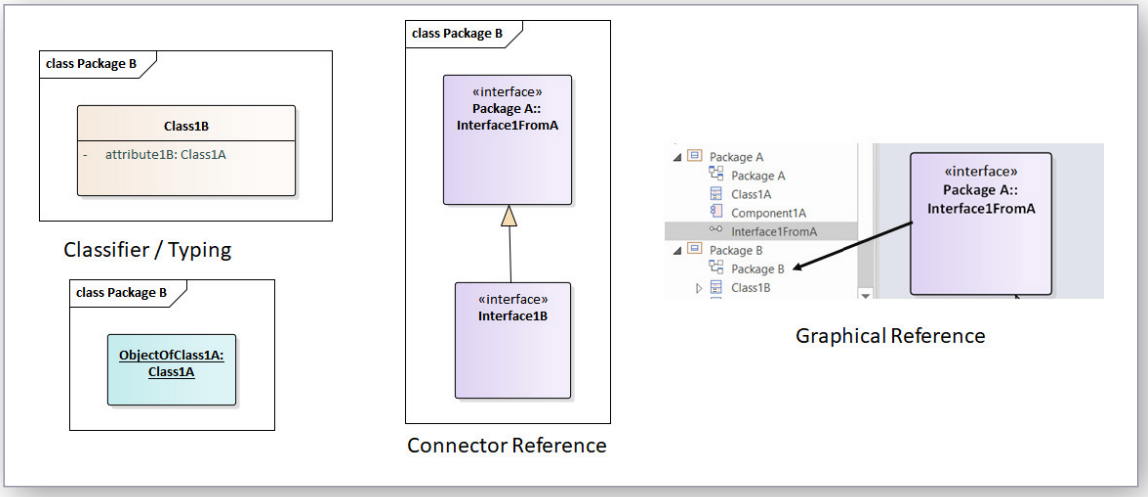


Abbildung 3: Arten von Referenzen bei Modellelementen

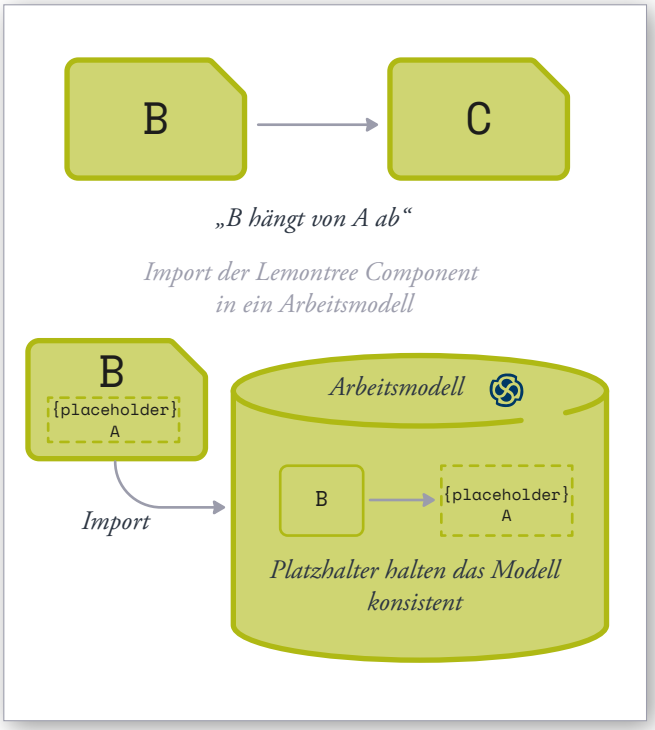


Abbildung 4: Import einer LemonTree Component mit Abhängigkeiten

In einem Modell sind Quelle und Ziel einer Referenz immer vorhanden. Zerschneidet man ein Modell in verschiedene Komponenten kann es vorkommen, dass Quelle und Ziel einer Referenz getrennt werden. Dadurch würde die Referenz verloren gehen, da im zerteilen Modell nur mehr eine Seite der Referenz vorhanden ist. Daher arbeiten LemonTree Components mit sogenannten Platzhaltern: Sie agieren als Stellvertreterelemente, um das fehlende Ende eine Referenz aufzulösen.

Durch den Mechanismus der Platzhalter entsteht bei jeder beliebigen Konstellation von Komponenten immer ein konsistentes Modell. Gewisse Abhängigkeiten können aber aufgrund von Modellierungsrichtlinien oder Prinzipien nicht erwünscht sein, wie z.B. zirkuläre Abhängigkeiten.

AUFLÖSEN VON ABHÄNGIGKEITEN

Ist eine bestimmte Konstellation von Abhängigkeiten nicht erwünscht, wird sie mit der Abhängigkeitsübersicht aufgelöst.

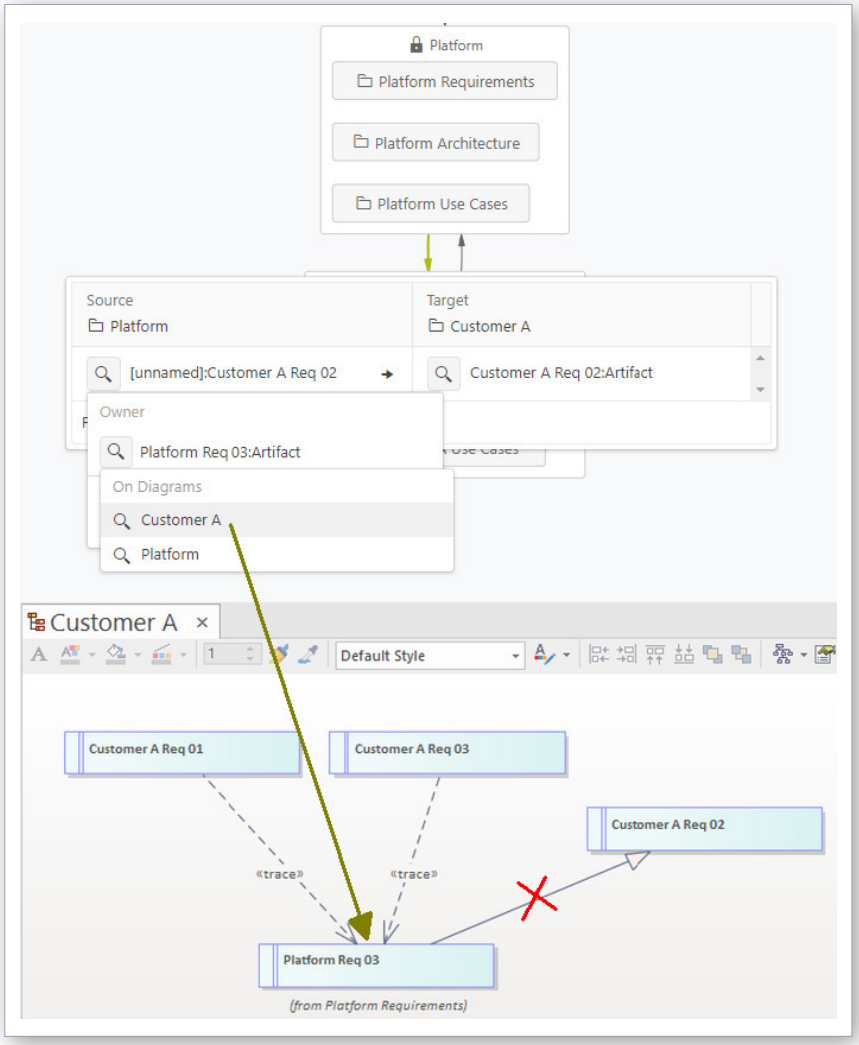


Abbildung 5: Auflösen einer zirkulären Abhängigkeit zwischen LemonTree Components

Im Beispiel in Abbildung 5 wird eine unerwünschte zirkuläre Abhängigkeit zwischen der Plattformkomponente und einer Projektkomponente aufgelöst. Hinter der zirkulären Abhängigkeit steckt eine irrtümlich erstelle Generalisierungsbeziehung zwischen einer Plattform und einer Kundenanforderung. LemonTree ermöglicht die Navigation zum Ursprung von Referenzen, um sie dann in Enterprise Architect aufzulösen. In Abbildung 5 betrifft das die Löschung der Generalisierungsbeziehung.

Wurden alle LemonTree Components spezifiziert und unerwünschte Abhängigkeiten aufgelöst, werden die Komponenten exportiert und in separaten Git Repositories abgelegt.

ARBEIT MIT LEMONTREE COMPONENTS IN VERSIONSKONTROLLE

Beim Export einer LemonTree Component wird eine MPMS (Model Package Management System) Datei erzeugt. Diese Datei basiert auf JSON und wird in das Git Repository eingecheckt und dort verwaltet. Da die MPMS Datei lediglich ein Extrakt der Komponente ist, wird für die Bearbeitung der Komponente mit LemonTree Automation ein Arbeitsmodell erzeugt. Diese wird im Gegensatz zu LemonTree Component nicht in der Versionskontrolle verwaltet.

ERZEUGUNG EINES TEMPORÄREN ARBEITSMODELLS

Zur Bearbeitung einer LemonTree Component wird ein temporäres Arbeitsmodell erzeugt, in das die Komponente (mit allen Abhängigkeiten) mittels LemonTree Automation importiert wird. Die Ablauflogik für die Erstellung des Arbeitsmodells sowie die Bereitstellung der referenzierten Abhängigkeiten wird von einem Skript (z.B. Powershell oder Python) übernommen. Das Skript kopiert ein leeres Enterprise Architect Modell (als Template) und importiert dorthin die zu bearbeitende LemonTree Component.

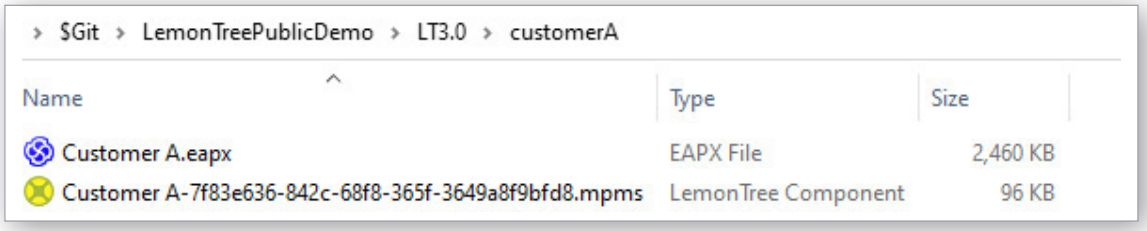


Abbildung 6:Arbeitsmodell für die LemonTree Component „Customer A“

Neben der primären LemonTree Component werden auch alle Komponenten importiert, zu denen eine ausgehende Referenz besteht. Ziel ist es, ein Arbeitsmodell ohne Platzhalter zu erzeugen. Die referenzierten Elemente werden als Read-Only Komponenten importiert um zu verhindern, dass sie in diesem Arbeitsmodell geändert werden.

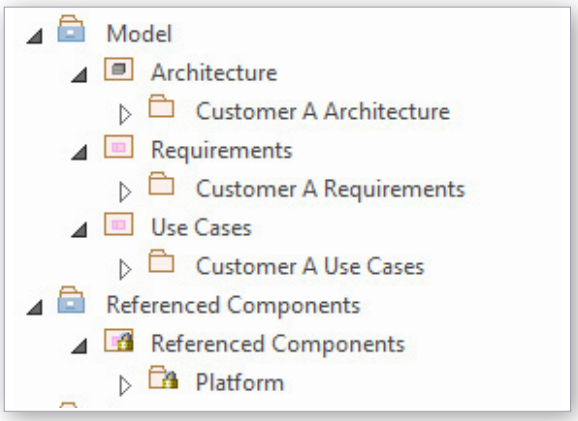


Abbildung 7: Importierte Komponente „Customer A“ mit schreibgeschützter Komponente „Platform“

Die Anpassung einer referenzierten Komponente muss im entsprechenden Git Repository geschehen. Im Anschluss wird ein Update der referenzierten Komponente importiert.

Da das Enterprise Architect Modell selbst nur temporär ist und nicht in der Versionskontrolle verwaltet wird, muss die LemonTree Component wieder in das MPMS Format exportiert werden, um sie schließlich mit einem Git Commit und Push zu veröffentlichen.

EXPORT DER GEÄNDERTEN LEMONTREE COMPONENT

Zur Veröffentlichung der Änderungen einer LemonTree Component wird der Inhalt des Arbeitsmodells exportiert. Dazu wird (wie beim Import) ein Skript verwendet, das LemonTree Automation steuert. Dabei wird die MPMS Datei im Git Repository mit dem Export der LemonTree Component aus dem Arbeitsmodell überschrieben. Die Änderungen werden dann über Git Commit und Push veröffentlicht.

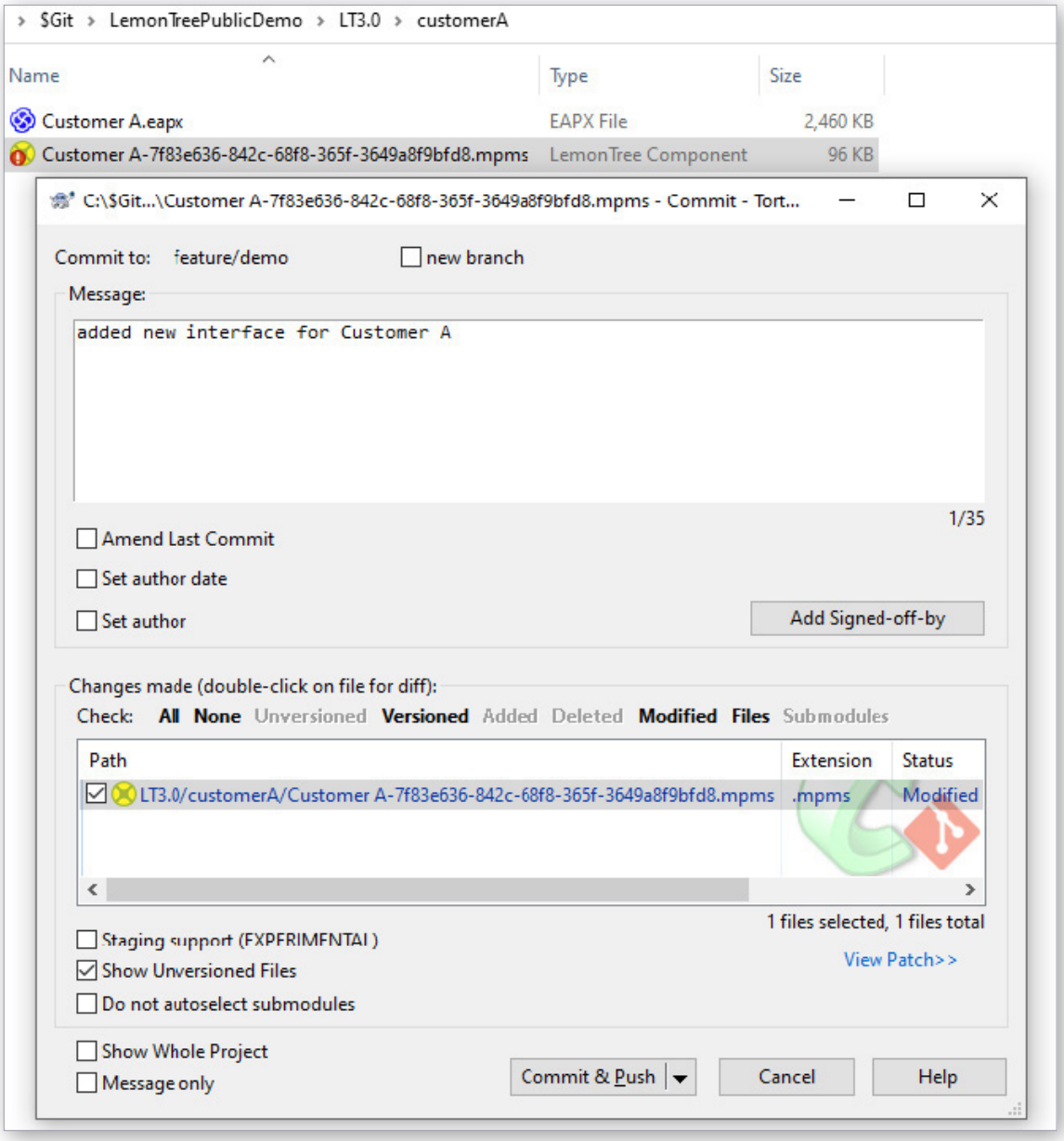


Abbildung 8: Veröffentlichung einer geänderten LemonTree Component in Git

Die Veröffentlichung der Änderung einer LemonTree Component mittels Git Push auf dem Git Server löst eine Reihe von automatisierten Aktionen auf einem Build-Server aus.

BUILD-SERVER PIPELINE

In diesem Dokument wird der Begriff „Build-Server Pipeline“ stellvertretend für verschiedenste Build-Server Systeme und Build-Prozesse - z.B. Azure DevOps, TeamCity, Jenkins, GitHub Actions - verwendet. In so einer Pipeline sind bestimmte Aktionen definiert, die im Kontext eines Versionskontroll-Repositories zur Ausführung kommen, sobald eine neue Revision im Git Repository (über Commit & Push) veröffentlicht wird. Abhängig von den unter Versionskontrolle stehenden Dateien werden unterschiedliche Aktionen ausgeführt.

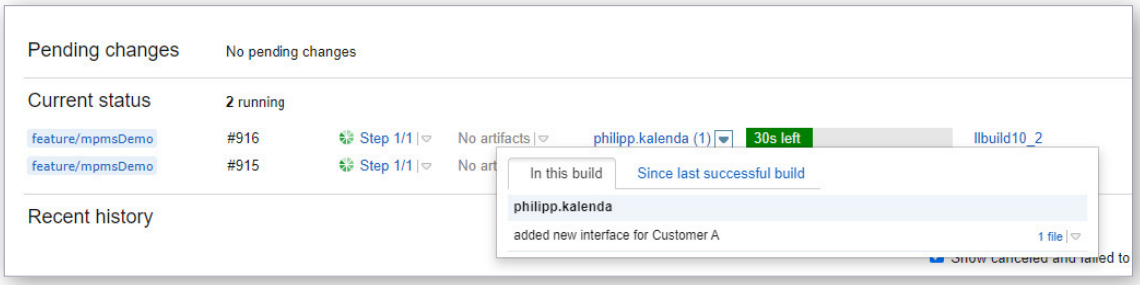


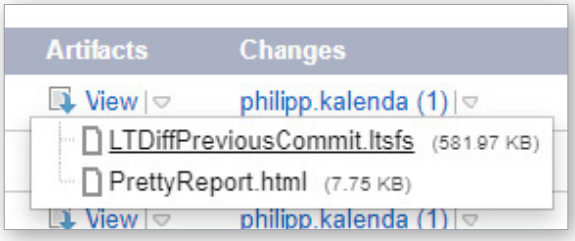
Abbildung 9: Build-Run nach Commit der MPMS Datei

Bei der Software-Entwicklung werden z.B. Projekte kompiliert, getestet und verifiziert. Im Fall von versionierten Modellen können ebenso Validierungen ablaufen (z.B. mittels SQL Queries im Enteprrise Architect Modell) oder es wird ein Vergleich von bestimmten Versionen mit Hilfe von LemonTree ausgeführt.

Für die Versionierung von LemonTree Components bieten sich Diff- und Merge-Operationen an.

AUTOMATISIERTER VERGLEICH VON LEMONTREE COMPONENTS

Um jede Änderung einfach nachverfolgen zu können, berechnet jeder ausgelöste Build-Run den Unterschied zwischen der aktuellen Version und der Vorgängerversion. Da bei einem Build-Run der Auslöser immer der aktuelle Commit ist, ist in diesem Kontext die LemonTree Component (MPMS Datei) des aktuellen Commit vorhanden. Zusätzlich dazu holt sich das Build-Skript die Vorgängerversion aus dem Git Repository und startet mit LemonTree Automation einen Vergleich. Da LemonTree Automation während des Build-Prozesses läuft und kein User Interface zur Verfügung stellt, erstellt LemonTree Automation eine sogenannte „Single File Session“. Dieses Artefakt beinhaltet die verglichenen Modelle bzw. LemonTree Components, damit deren Vergleich jederzeit reproduziert werden kann. Dieses Session Artefakt wird am Ende des Build-Runs als Ergebnis veröffentlicht und ist so für jede Person mit Zugriff auf das Build-Projekt einsehbar.



Die LemonTree Single File Session kann heruntergeladen und geöffnet werden, um den Vergleich mit dem letzten Commit der LemonTree Component zu sehen.

Abbildung 10: Erzeugte Diff Session als Build-Artefakt

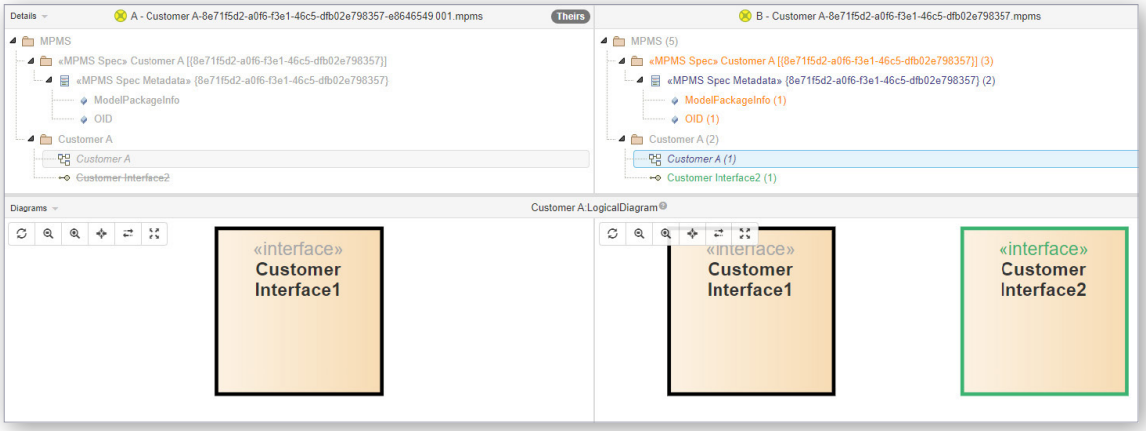


Abbildung 11: Diff einer LemonTree Component aus LemonTree Single File Session

Neben automatisierten Vergleichen wird auch die Berechnung von möglichen Konflikten bei Merge-Operationen unterstützt. Dies ist speziell notwendig, wenn mit einer Multi-Branch Strategie (z.B. mit GitFlow) gearbeitet wird.

MERGE PREVIEW VON LEMONTREE COMPONENTS (MULTI-BRANCH STRATEGIE)

Bei der Arbeit mit einer Multi-Branch Strategie (z.B. GitFlow) werden erstellte Branches wieder zusammengeführt. Dies passiert z.B. wenn ein Feature abgeschlossen oder ein Release veröffentlicht wird. Bevor jedoch eine Änderung übernommen bzw. in einen Branch integriert wird, wird zuerst ein Merge- bzw. Pull-Request erstellt. Dabei wird festgelegt, welche Änderung in welchem Branch zusammengeführt wird. Um zu beurteilen ob eine Änderung übernommen werden kann bzw. um diese freizugeben wird ein Review der Änderungen aus dem Branch durchgeführt. Um die Modelländerungen besser nachzuvollziehen wird auch in diesem Fall eine LemonTree Single File Session erzeugt. Dabei wird ein Vergleich zwischen dem Merge Target (z.B. der develop Branch) und dem aktuellen Branch berechnet (z.B. Feature Branch).

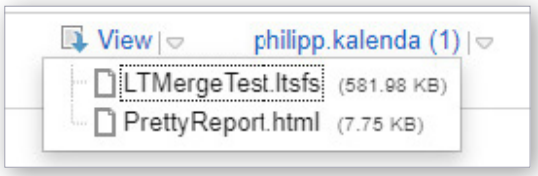


Abbildung 12: Merge Preview Artefakt

Dieses Merge Artefakt zeigt in einer LemonTree Session, wie das Endergebnis der LemonTree Component nach der Zusammenführung aussehen würde. Kommt es im Kontext des Merge-Vorgangs zu einem Konflikt, lässt sich dieser nicht automatisch auflösen. In diesem Fall wird eine Diff-Session erzeugt und der Build-Prozess bricht mit einer Fehlermeldung ab.

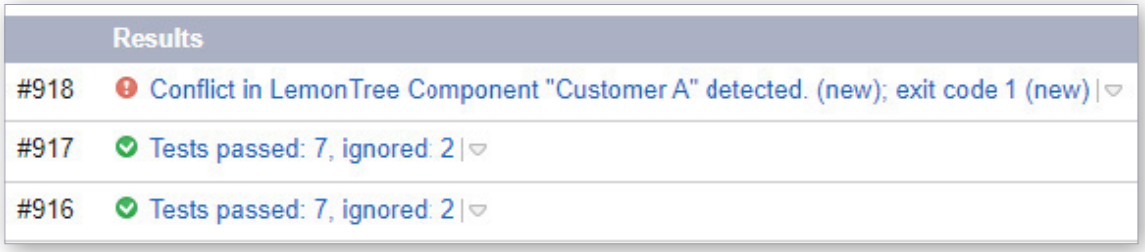


Abbildung 13: Fehlgeschlagener Build aufgrund eines Konflikts

Wird z.B. ein Feature abgeschlossen und ein Branch zusammengeführt, existiert nach dem erneuten Build-Vorgang eine neue Version der Komponente. Um diese Komponente im Gesamtkontext zu integrieren wird regelmäßig ein Gesamtmodell erstellt bzw. aktualisiert, in welchem alle vorhanden LemonTree Komponenten zusammengeführt werden.

INTEGRATION IN EIN GESAMTMODELL

Das Gesamtmodell wird üblicherweise an einer zentralen Stelle - etwa einem Fileshare oder einem Datenbank-Server - verwaltet. Es repräsentiert die Gesamtheit aller Komponenten in bestimmten Versionsständen. Um regelmäßig Updates einzuspielen, wird das Gesamtmodell abhängig von den Build-Prozessen der LemonTree Components aktualisiert.

Wird z.B. ein Pull Request einer Komponente abgeschlossen, werden die Änderungen aus dem Feature Branch nach develop zurück gemerged. Dieser Merge löst einen neuen Build im develop Branch aus, der das Gesamtmodell aktualisiert und die neueste Version der entsprechenden Komponente einspielt. Das Ergebnis des Build ist eine LemonTree Session für den Vergleich zwischen dem aktuellen develop Commit und dessen Vorgänger sowie ein Gesamtmodell mit dem aktuellsten Stand der LemonTree Component.

Das Gesamtmodell dient lediglich als Informationsquelle bzw. als Referenz, um die Kompatibilität der Komponenten untereinander zu prüfen. Die Änderungen der Komponenten erfolgen immer im jeweiligen Git Repository, unter Verwendung von LemonTree Component, LemonTree Automation und dem erzeugten Arbeitsmodell.

VERSCHIEDENE AUSBAUSTUFEN DES BEISPIEL-WORKFLOWS

Der oben beschriebene Workflow gibt ein Beispiel, wie mit LemonTree Components und Automation gearbeitet werden kann. Gezeigt wird dabei die aktuell maximale Ausbaustufe von LemonTree Tools im Kontext einer Build-Pipeline. Es ist jedoch nicht zwingend notwendig, diese umfassende Form der Toolkette aufzusetzen. Daher beschreiben wir nun die verschiedenen Ausbaustufen des Beispiel-Workflows. So wird deutlich, dass diese Arbeitsweise Schritt für Schritt bzw. gemäß Ihrer Anforderungen eingeführt werden kann.

Der gesamte Beispiel-Workflow sieht folgendermaßen aus:

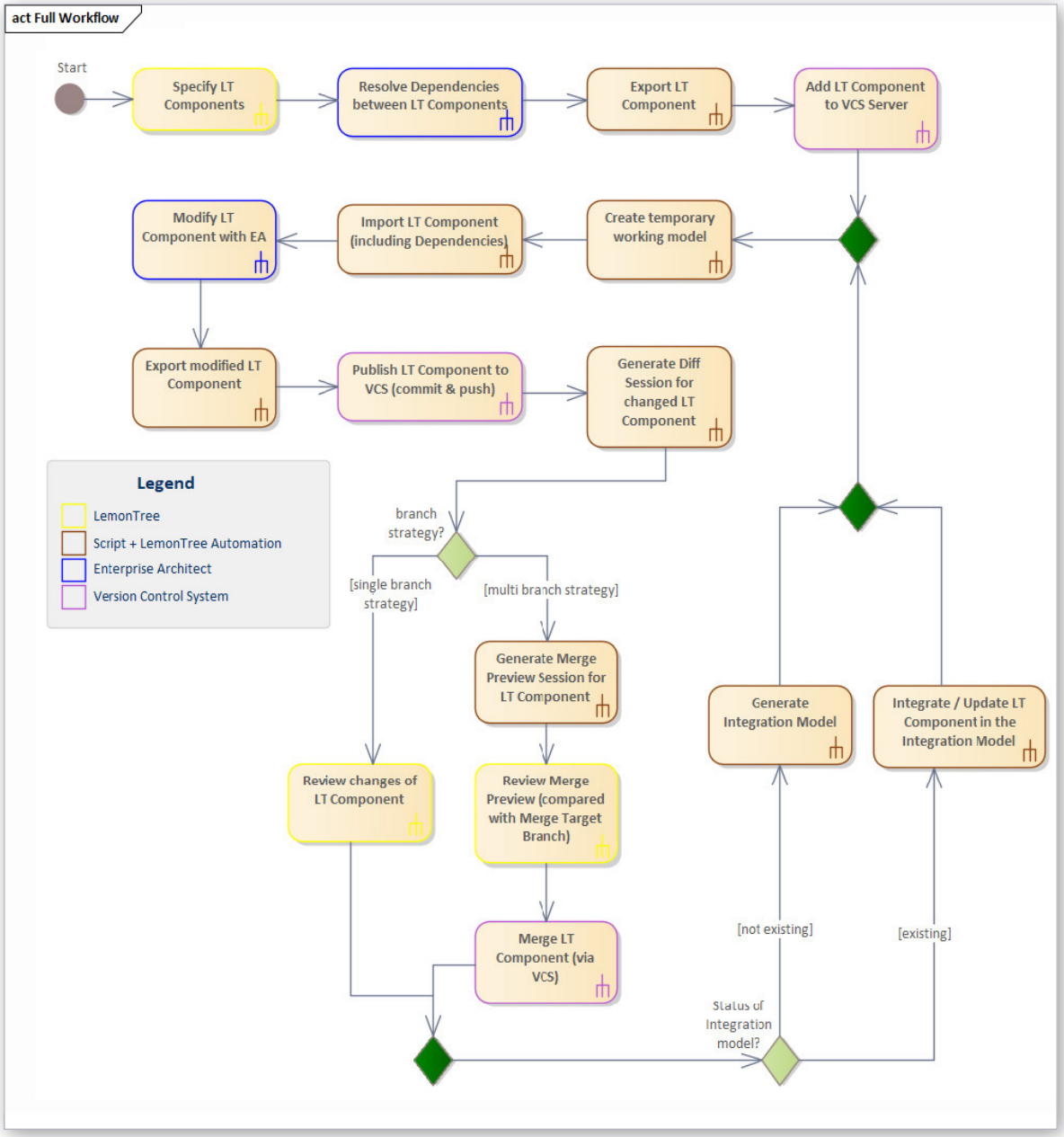


Abbildung 14: Gesamter Beispiel-Workflow

VARIANTE OHNE LEMONTREE COMPONENTS
(gesamtes Enterprise Architect Modell wird versioniert)

Bei dieser Variante wird nicht mit LemonTree Components gearbeitet, sondern mit einem gesamten Enterprise Architect Modell (eap, eapx oder qeax Datei). Sie bietet sich für kleine und mittelgroße Modelle an und auch dann, wenn sich in einem Modell keine Komponenten befinden, die in einem eigenen Lebenszyklus verwaltet werden sollen.

Bei dieser Variante entfallen die Schritte „Spezifikation von LemonTree Components“, „Arbeit mit LemonTree Components in Versionskontrolle“ und „Integration in ein Gesamtmodell“. Die Diff Sessions und Merge Previews - beschrieben im Kapitel „Build-Server Pipeline“ - können auch mit einem vollständigen Enterprise Architect Modell auf dem Build-Server erzeugt werden.

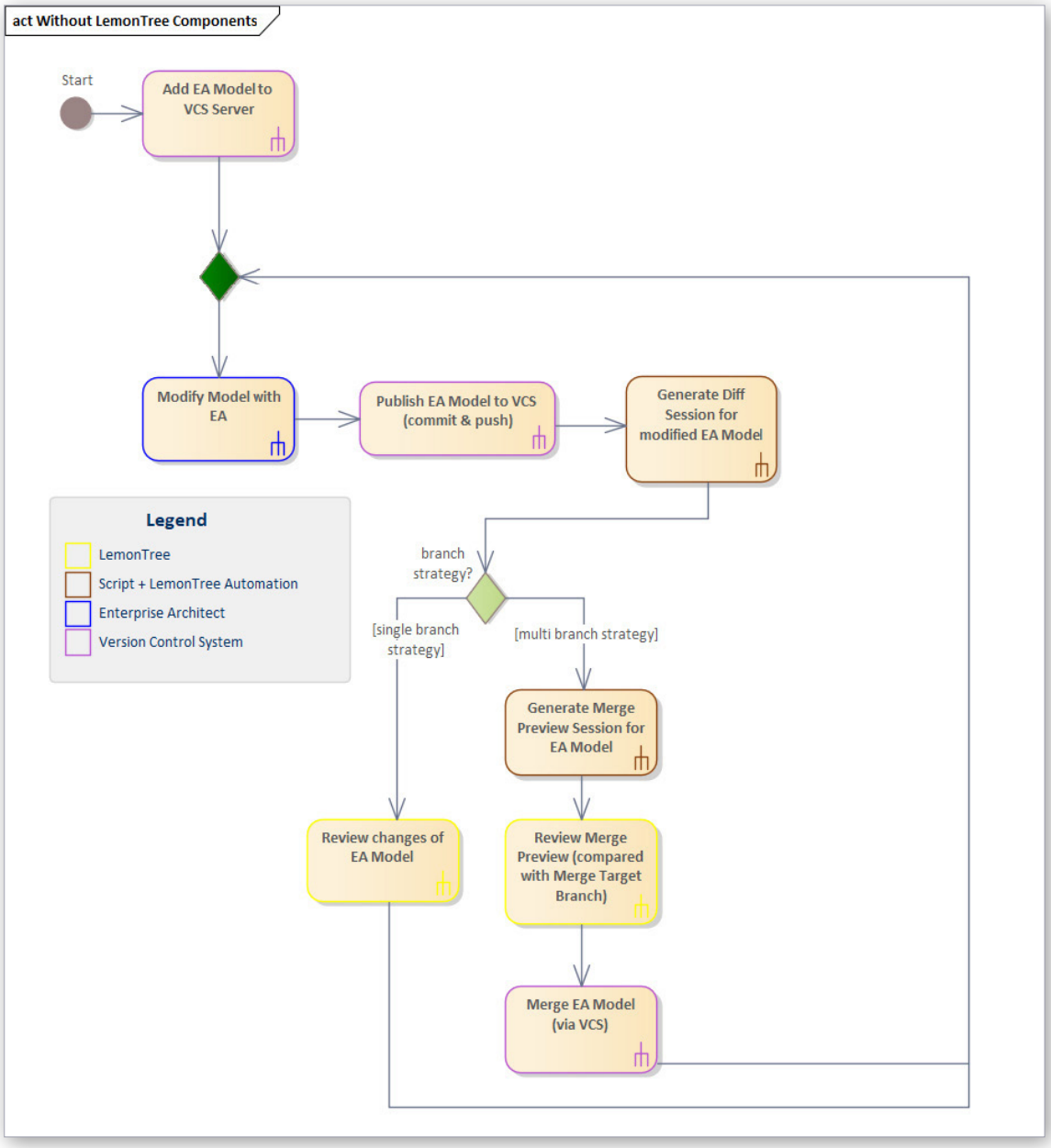


Abbildung 15: Workflow mit EA Model unter Versionskontrolle

VARIANTEN MIT LEMONTREE AUTOMATION IN VERSCHIEDENEN AUSBAUSTUFEN

STUFE 1: OHNE LEMONTREE AUTOMATION

Der geschilderte Workflow ist auch ohne Einsatz von LemonTree Automation möglich. Dabei werden jedoch sowohl der Austausch von LemonTree Components als auch die Vergleiche von Änderungen zu manuellen Aktivitäten. Es wird ohne Buildserver gearbeitet und die Aktualisierung des Integrationsmodells mit allen LemonTree Components erfolgt manuell über das LemonTree Addin.

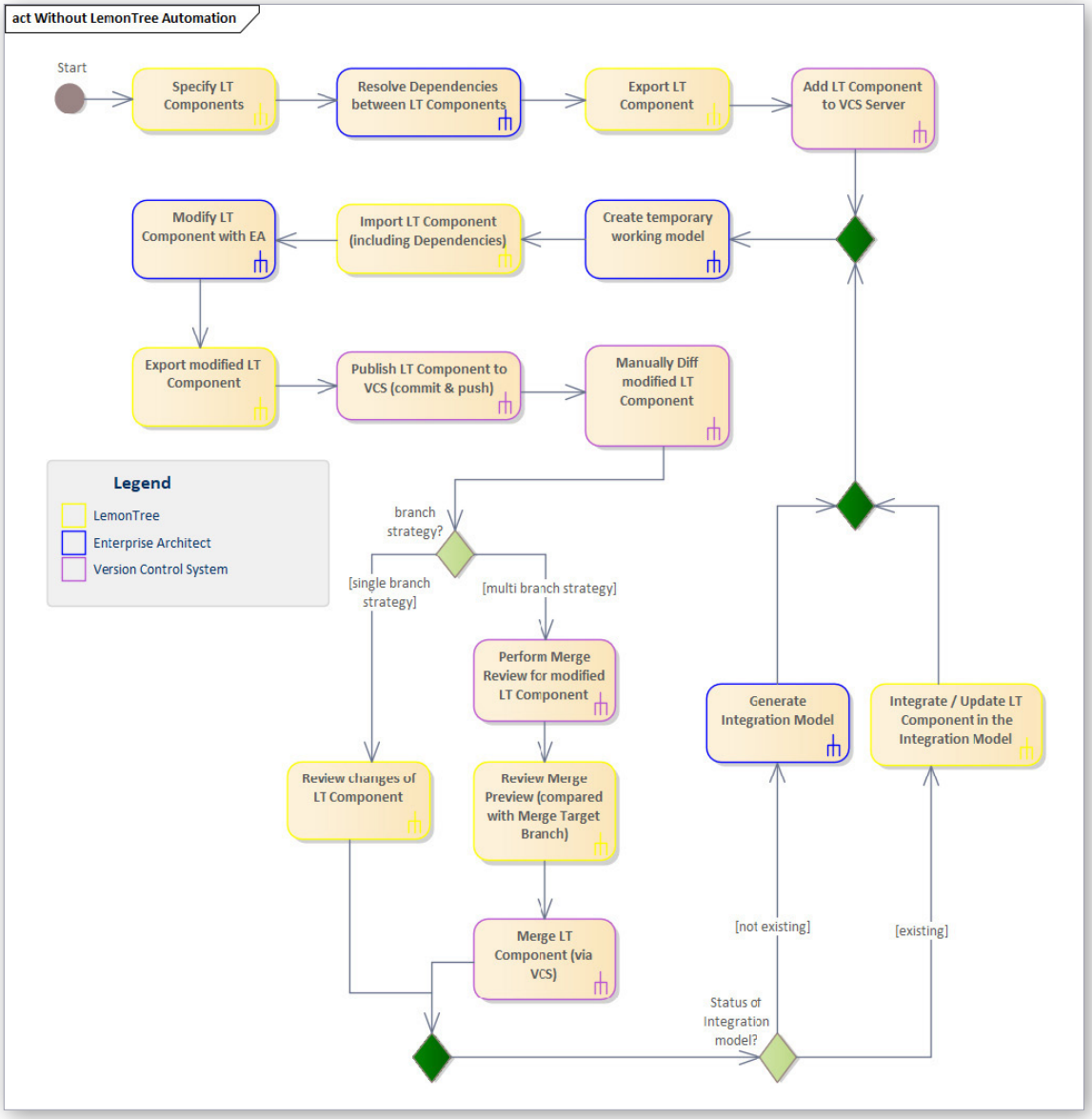


Abbildung 16: manueller Workflow ohne LemonTree Automation

STUFE 2: AUTOMATISIERTE ERSTELLUNG DES ARBEITSMODELLS
(ohne Build-Server)

In dieser Ausbaustufe dient LemonTree Automation zur Erstellung des temporären Arbeitsmodells unter Einbeziehung der abhängigen LemonTree Components. Außerdem werden mit Hilfe eines Skripts die geänderten LemonTree Components aus dem Arbeitsmodell extrahiert, um sie dann in der Versionskontrolle zu veröffentlichen. Sie benötigen keinen Buildserver und die Aktualisierung des Integrationsmodells mit allen LemonTree Components erfolgt manuell über das LemonTree Addin.

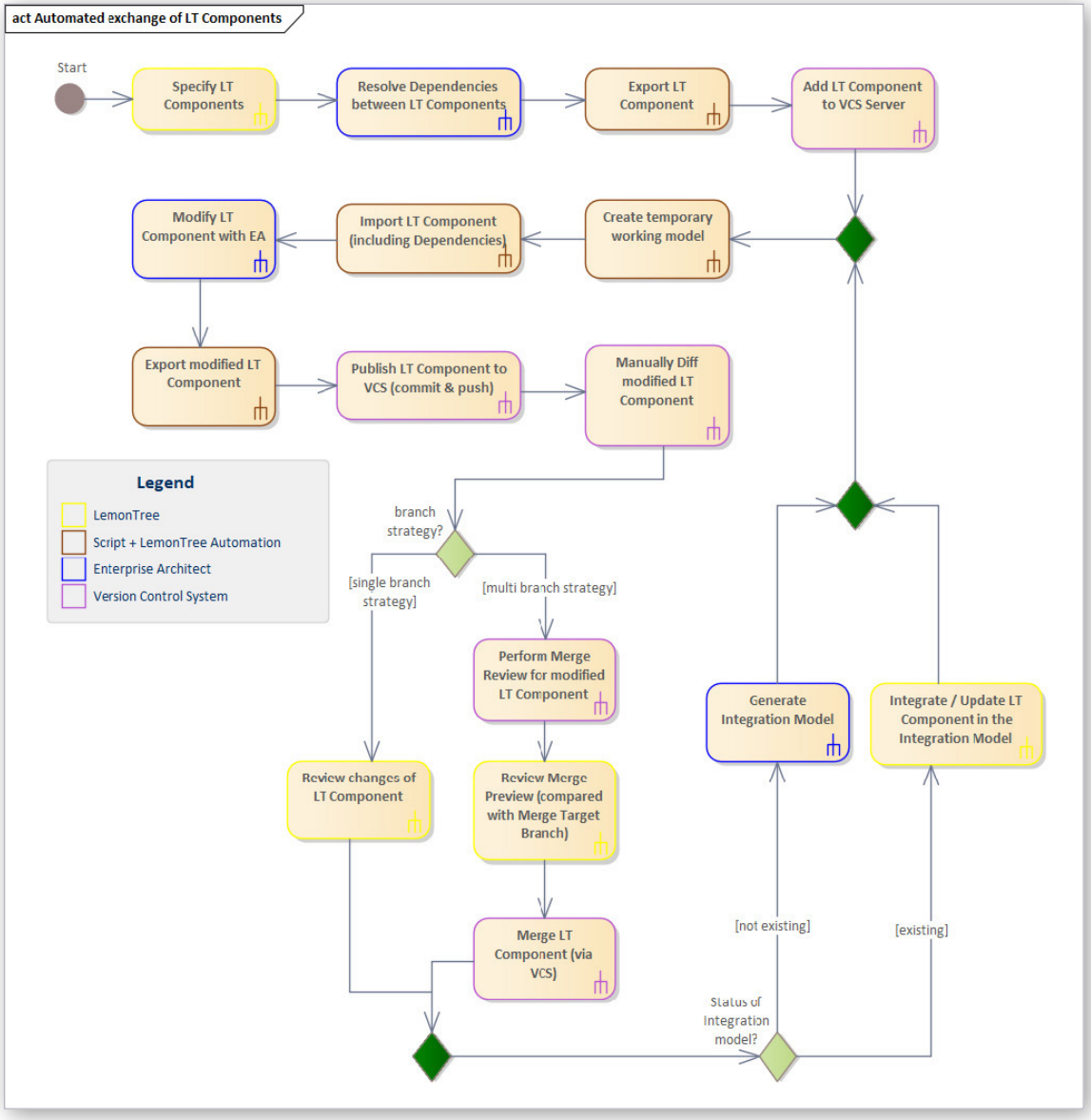


Abbildung 17: Automatisierter Austausch von LemonTree Components

STUFE 3: DIFF / MERGE REPORT AM BUILD-SERVER

In Stufe 3 wird zusätzlich ein Build-Server verwendet, um automatisierte Diff/Merge-Reports zu erstellen. Die Aktualisierung des Integrationsmodells mit allen LemonTree Components erfolgt manuell über das LemonTree Addin.

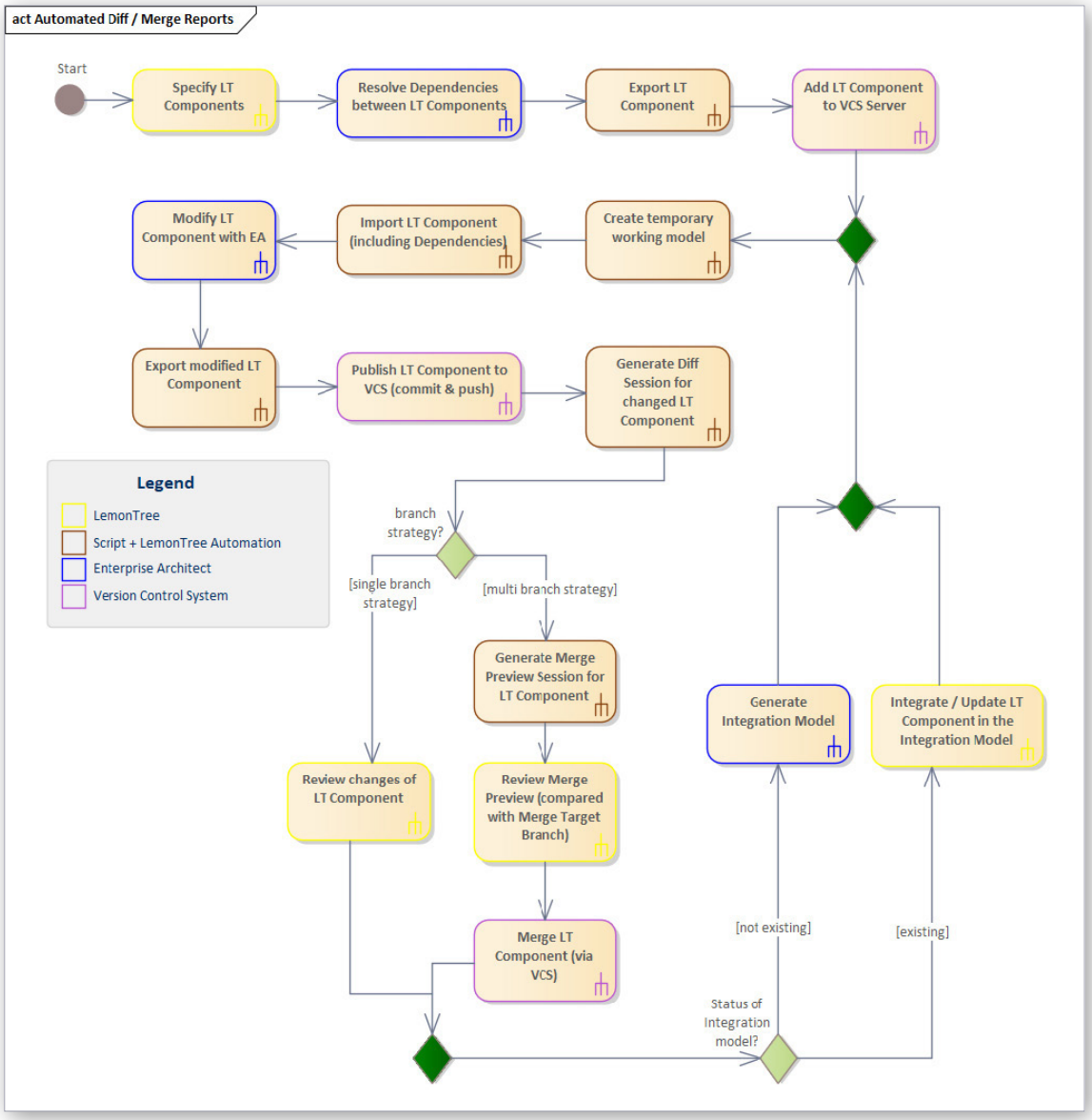


Abbildung 18: Automatisierte Erstellung von Diff/Merge-Reports

STUFE 4: INTEGRATION IN EIN (ZENTRALES) GESAMTMODELL

In der höchsten Ausbaustufe (Stufe 4) aktualisiert sich das Integrationsmodell automatisch. Dafür wird das zentrale Modell (z.B. Fileserver oder Datenbank) über ein Skript und das LemonTree Component Update durch LemonTree Automation aktuell gehalten. Diese Workflow-Erweiterung entspricht dem umfassenden Workflow aus Abbildung 14.

CONCLUSIO

Die Arbeit mit LemonTree Automation und LemonTree Components ermöglicht eine feingranulare Versionierung von Teilen eines Gesamtmodells. Dabei werden gewohnte Prozesse und Tools aus dem Bereich Continuous Integration und DevOps wiederverwendet. LemonTree Automation lässt sich als Werkzeug nahtlos mit anderen gängigen Tools aus der Entwicklung kombinieren, um in gewohnten Build-Prozessen neben Code auch Modelle zu berücksichtigen.

https://www.lieberlieber.com/lemontree/de/automation/

Die Autoren

Philipp Kalenda
Senior Consultant

Dr. Konrad Wieland
CEO

Kontaktieren Sie uns bitte unter
welcome@lieberlieber.com

Für den Inhalt verantwortlich

LieberLieber

LieberLieber Software
Handelskai 340, Top 5
1020 Vienna, Austria
+43 662 90600 2017