CONTINUOUS INTEGRATION MIT **ENTERPRISE ARCHITECT**

ZUSAMMENSPIEL VON LEMONTREE, GIT, PROCLOUDSERVER UND PROLABORATE

LieberLieber Software GmbH

EINLEITUNG

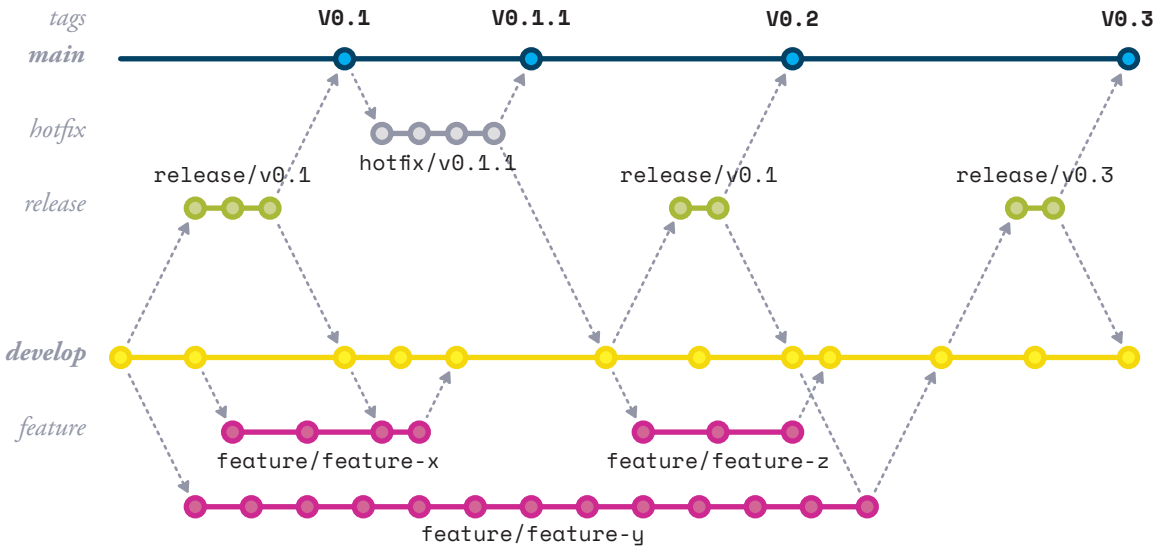
Im Zuge der zunehmend agilen Ausrichtung in der Software-Entwicklung fragen uns unsere Kunden immer wieder, ob Modellierung und Agilität gut zusammenpassen. Unsere Meinung dazu ist ganz klar und wurde auch in verschiedenen Studien belegt: Nur mit einer richtig angewandten Modellierung lassen sich agile Prozesse angesichts zunehmender Komplexität überhaupt umsetzen, wenn man dabei auch alle Vorschriften und Anforderungen nachvollziehbar dokumentieren will. Als Beispiel wollen wir uns in diesem Text mit „Continuous Integration“ befassen, die in der klassischen Software-Entwicklung seit einiger Zeit state of the art ist. Ziel dabei ist es, das zu entwickelnde Software-Produkt kontinuierlich zu kompilieren, zu testen und zusammenzuführen. Zusätzlich wird regelmäßig der Stand der Entwicklung veröffentlicht: intern ein stabiler Entwicklungsstand, extern ein offizieller Release der Software.

Wie bringt man nun aber „Continuous Integration“ in die modellbasierte Softwareentwicklung? In der Werkzeug-Kette rund um Enterprise Architect (LemonTree, Git, Pro Cloud Server, Prolaborate) finden sich aktuell alle Voraussetzungen, um dieses Szenario zu verwirklichen. Aus unserer Erfahrung sind diese Möglichkeiten am Markt derzeit einzigartig und sie eröffnen unseren Kunden ganz neue Möglichkeiten. Daher wollen wir Ihnen vorstellen, welche Vorgehensweisen der klassischen Software-Entwicklung für ein „Continuous Modeling“ mit dieser Werkzeugkette übernommen werden können. Dabei wird auch klar, dass die Arbeit mit einer zentralen Datenbank (Prolaborate) und der Umgang mit LemonTree/Git kein Widerspruch sind, sondern sich sehr gut parallel einsetzen lassen.

KOLLABORATIVE MODELLIERUNG **MIT GIT**

Mit LemonTree von LieberLieber ist es bereits möglich, eine etablierte Vorgehensweise aus der Software-Entwicklung in die Modellierung zu übernehmen, nämlich die Versionierung von Modellartefakten. Bei der täglichen Arbeit am Modell empfehlen wir dazu einen Versionskontrollansatz mit Git. Git ermöglicht paralleles und verteiltes Arbeiten an einem Modell. Allem voran ist ein Vorteil bei diesem Ansatz, dass jeder User auf eine lokale Kopie des geteilten Modells zugreift. Daher ist keine Netzwerkverbindung zu einem „shared model" notwendig und es entfallen eventuelle Wartzeiten auf Grund von Netzwerklatenz. So wird es erheblich einfacher, zügig Änderungen durchzuführen oder einen schnellen Blick in das Modell zu werfen.

Wird ein Modell erstellt, so bietet sich der Workflow „GitFlow" für die Kollaboration an. Dabei wird nie auf dem Hauptstrang „develop" (gelber Strang) direkt entwickelt, sondern immer in eigenen, abgegrenzten „feature branches" (rote Stränge).



Erst wenn ein Feature fertig modelliert und geprüft wurde, wird dieses abgeschlossen und zurück in den develop branch gebracht. Der Vorteil dieses Ansatzes ist es, dass abgekapselt von anderen Entwicklungen Änderungen durchgeführt werden können, ohne laufend neue Änderungen integrieren zu müssen. Erst beim Abschluss eines Features werden die zwischenzeitlichen Änderungen aus develop relevant und es kann bewusst ein Merge der Modelle mittels LemonTree durchgeführt werden.

Der Release-Prozess und die finale Veröffentlichung des Modells werden ebenfalls von GitFlow unterstützt. Über einen „release branch" wird ein Stand aus dem develop als Release-Kandidat deklariert. Hier können noch letzte Änderungen gemacht und eventuelle Fehler behoben werden. Ist ein Release-Kandidat freigegeben, wird der Stand des Modells in den Hauptstrang, den sogenannten „master", übertragen. Dort wird mittels eines „Tags" der veröffentliche Stand mit einem Namen versehen und eingefroren.

Kollaborative Modellierung mit Git

Diverse Normen fordern den Einsatz eines entsprechenden Configuration Managements und das bezieht sich auf alle Elemente – somit auch auf Modelle. Mit dem oben genannten Ansatz können parallel mehrere Features modelliert und veröffentlichte Stände gesichert werden. Es ist außerdem möglich, historische Stände des Models mit z.B. dem aktuellen Release zu vergleichen um herauszufinden, was sich über die Zeit verändert hat.

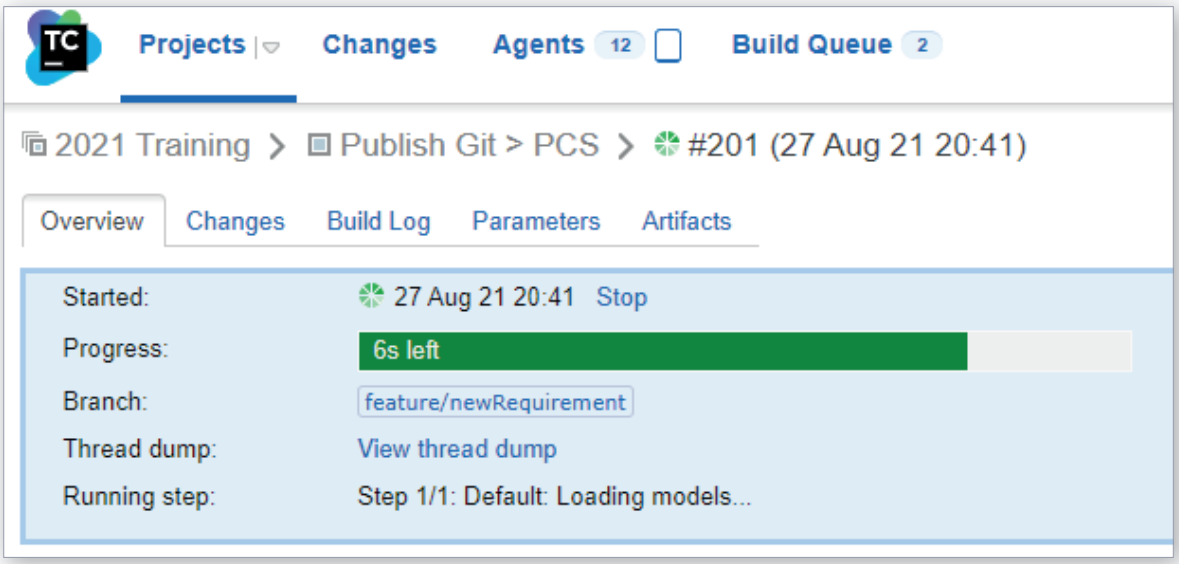
Graph

In der Grafik sehen Sie ein Beispiel aus dem Tool SmartGit. Hier wurde ein Release-Kandidat „1.1“ definiert. Ein nicht abgeschlossenes Feature für ein Architektur-Refactoring sowie ein in der Zwischenzeit abgeschlossenes Feature „Safety Control“ haben es nicht in den Release geschafft.

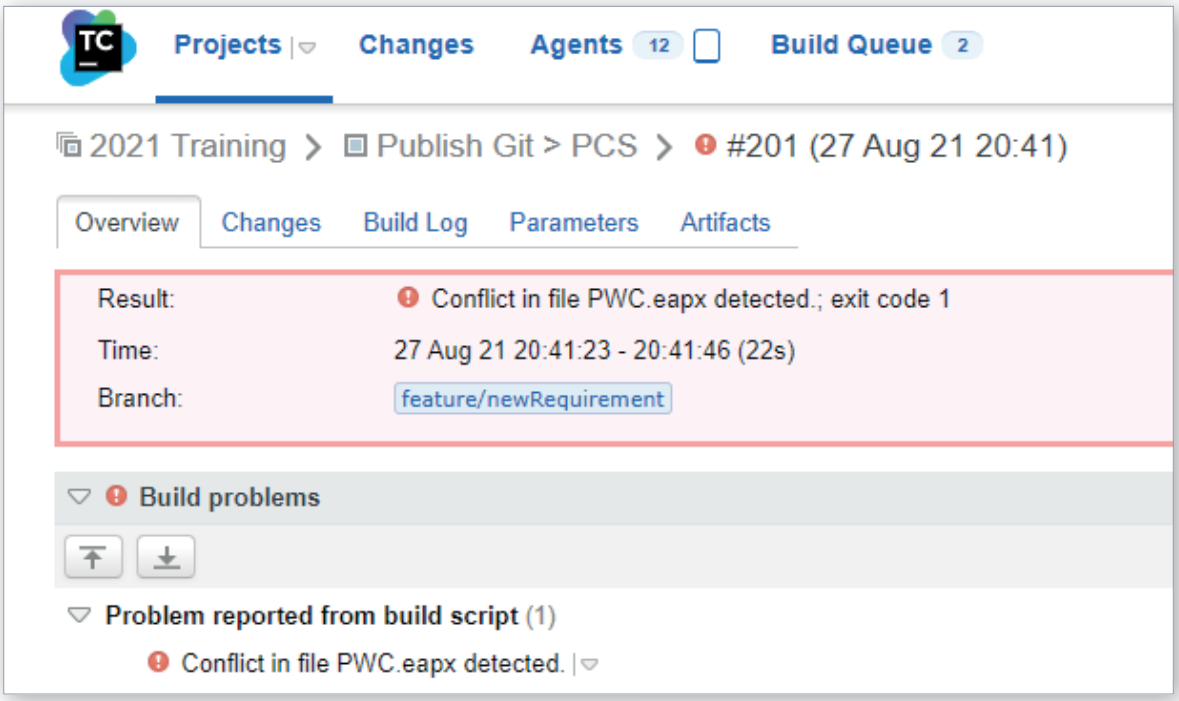
Wird eine Merge Operation eines Modells durchgeführt, versucht LemonTree das Modell automatisch zusammenzuführen. Dies ist nicht der Fall, wenn ein Konflikt auf Elementebene existiert. Das kommt vor, wenn z.B. die gleiche Eigenschaft des gleichen Elements in beiden Versionen unterschiedlich geändert wurde. Um den Konfliktfall vorzeitigt und nicht erst beim Integrieren des Features zu erkennen, kann die spezielle Automation Edition von LemonTree verwendet werden.

CONTINUOUS INTEGRATION

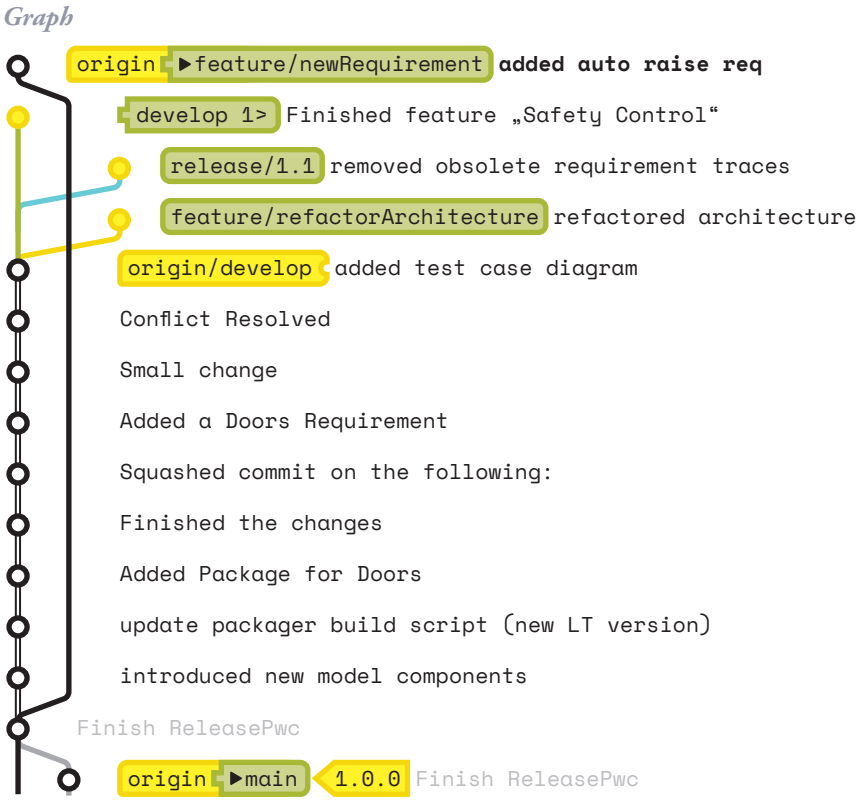
Wenn in einem feature branch gearbeitet wird sind jene Änderung von Interesse, die zu einem Konflikt gegenüber dem develop branch führen. Um das automatisch zu erkennen, lässt sich ein Build-Server, wie z.B. TeamCity oder Jenkins verwenden. Ähnlich wie bei der Software-Entwicklung, löst jeder Commit und Push in einem Repository einen neuen „Build“ aus:



Im Gegensatz zur Software kann ein Modell aber nicht kompiliert und auf etwaige Syntaxfehler geprüft werden. Mit einem automatisierten Modellvergleich lässt sich feststellen, ob ein feature branch im Konflikt zum develop branch steht. Ist das der Fall, wird der Build abgebrochen und eine Fehlermeldung ausgegeben:

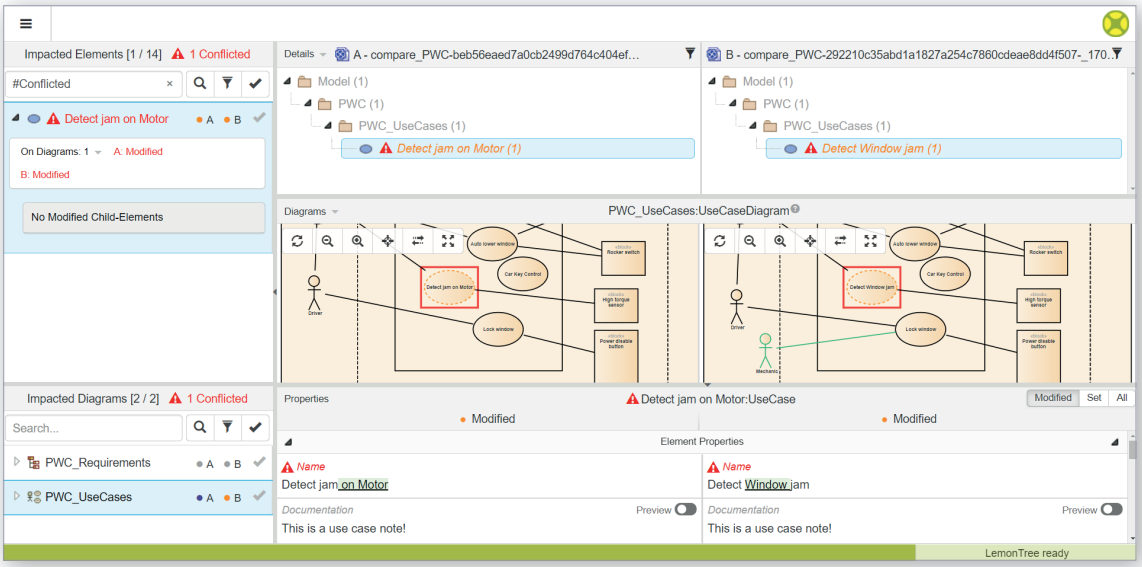


Es können nun in Git die zwei branches (feature/newRequirement und develop) miteinander verglichen werden:



Dabei ist auch zu erkennen, dass das Feature „newRequirement“ einen älteren Ursprung hat und sich der develop Branch maßgeblich weiterentwickelt hat.

LemonTree zeigt nun, welche Änderungen zu einem Konflikt führen:



Damit kann der Modellierer vorausschauend arbeiten und den Konflikt bei sich im Modell auflösen, bevor er einen Antrag für die Fertigstellung seines Feature (einen sogenannten Merge- oder Pull-Request) stellt.

Bis zu diesem Zeitpunkt ist das Modell ausschließlich im Versionskontroll-Repository verfügbar. Oftmals ist es aber notwendig, das Modell automatisiert zur Verfügung zu stellen, damit Außenstehende das Modell als Wissensdatenbank nutzen können.

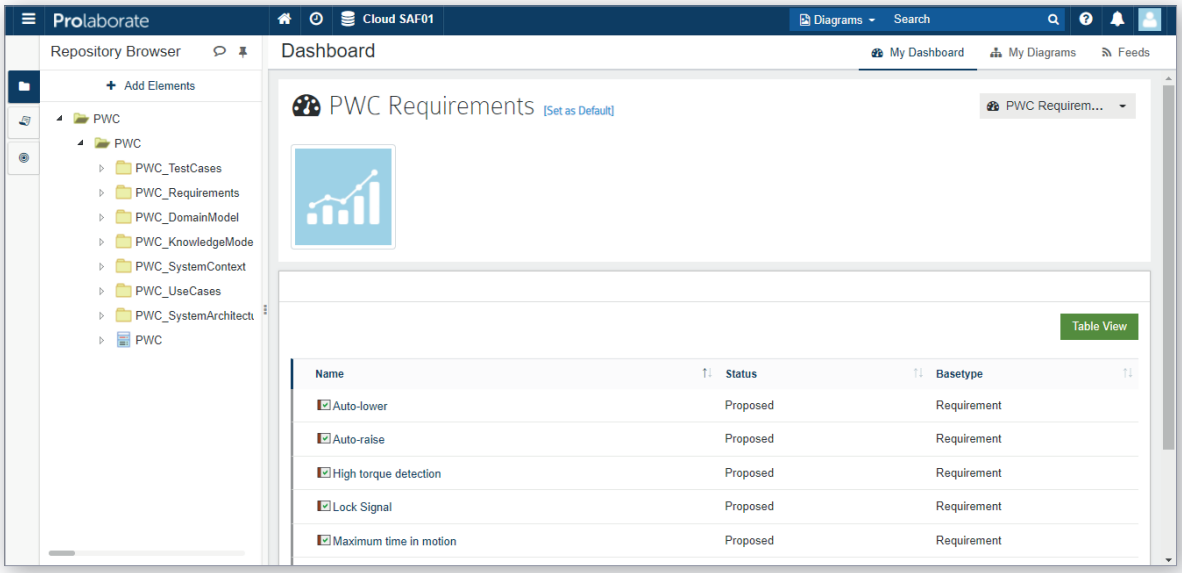
MODEL-REVIEW MIT **PROLABORATE**

Um neben Continuous Integration auch Continuous Delivery / Deployment zu betreiben, kann die oben bereits erwähnte Build-Server Struktur verwendet werden. Damit das Modell jedoch auch außerhalb der Versionskontrolle zugänglich bleibt, braucht es eine einfache Zugriffsmöglichkeit auf Modelle. Dafür bietet sich ein Web-Browser an, der den Inhalt eines geteilten Modell-Repositories anzeigt. Um das Modell außerhalb der Versionskontrolle zu hosten, wird der Pro Cloud Server von Sparx Systems verwendet.

Ähnlich wie bei der oben beschriebenen Continuous Integration wird hier ebenfalls bei jedem Build ein Automatismus gestartet. Ein Project Transfer überträgt dabei das lokale Modell aus der Versionskontrolle in ein Datenbank-Repository, welches als Datenquelle für den Pro Cloud Server und Prolaborate dient. Dabei wird unterschieden, welcher Branch aktuell gebaut wird. Für den develop und den master branch gibt es eigene Cloud Repositories, die mit unterschiedlichen Benutzern geteilt werden. Jene Benutzer, die regelmäßig Reviews durchzuführen, haben Zugriff auf das Entwicklungsmodell, also das develop Repository. Außenstehende oder Konsumenten des Modells sind nur berechtigt, finalisierte Stände des Modells zu sehen und haben daher nur Zugriff auf das master Repository.

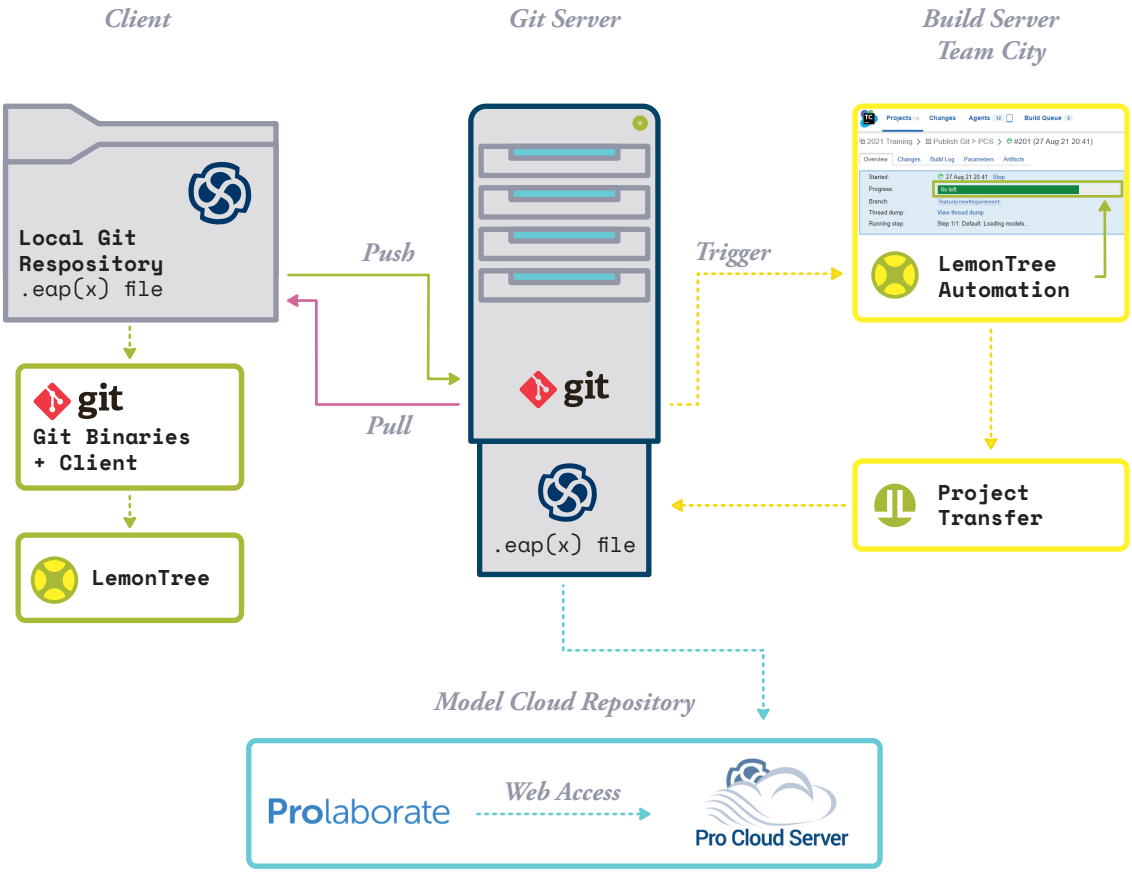
Um den Zugriff auf das Modell zu vereinfachen, lassen sich die Webschnittstellen des Pro Cloud Server verwenden.

Prolaborate bietet neben der Anzeige des Modells im Browser auch noch die Möglichkeiten von Dashboards und Reviews. So lassen sich intuitive und personalisierte Dashboards entwerfen. Sobald sich ein Benutzer anmeldet, landet er direkt auf den dynamischen Dashboards, die speziell für ihn entworfen wurden. Alle benötigten Architekturinformationen werden dabei auf einer einzigen Seite präsentiert. Für die Durchführung von Reviews ermöglicht Prolaborate einen agilen Überprüfungsprozess, indem jeweils die richtigen Teams mit der Durchführung der detaillierten Überprüfungen beauftragt werden. So lassen sich die Ergebnisse aus dem Enterprise Architect überprüfen, ohne dass dafür zusätzliche Dokumente erstellt werden müssen.



ZUSAMMENFASSUNG

Abschließend wird ein Überblick des gesamten „Continuous Modeling“ Workflows gezeigt.



Für den Inhalt verantwortlich

LieberLieber

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

Philipp Kalenda
Senior Consultant

Dr. Konrad Wieland
CEO

Kontaktieren Sie uns unter
welcome@lieberlieber.com