# VERSIONEN UND VARIANTEN BEHERRSCHEN

## EIN EINHEITLICHER ANSATZ FÜR KOMPLEXES SYSTEMS ENGINEERING

Versionierung und Variantenhandling sind im Engineering komplexer Systeme unverzichtbar, um Qualität, Effizienz und Nachvollziehbarkeit zu gewährleisten und die Herausforderungen von Komplexität und Vielfalt effektiv zu bewältigen. Leider ist es aber in der Praxis meist so, dass man e ntweder ein gutes Versionierungskonzept einführt oder versucht, Varianten zu verwalten. Nun ist es LieberLieber gelungen, eine effektive Lösung für Versions- und Variantenmanagement zu entwickeln.

*LieberLieber Software GmbH*

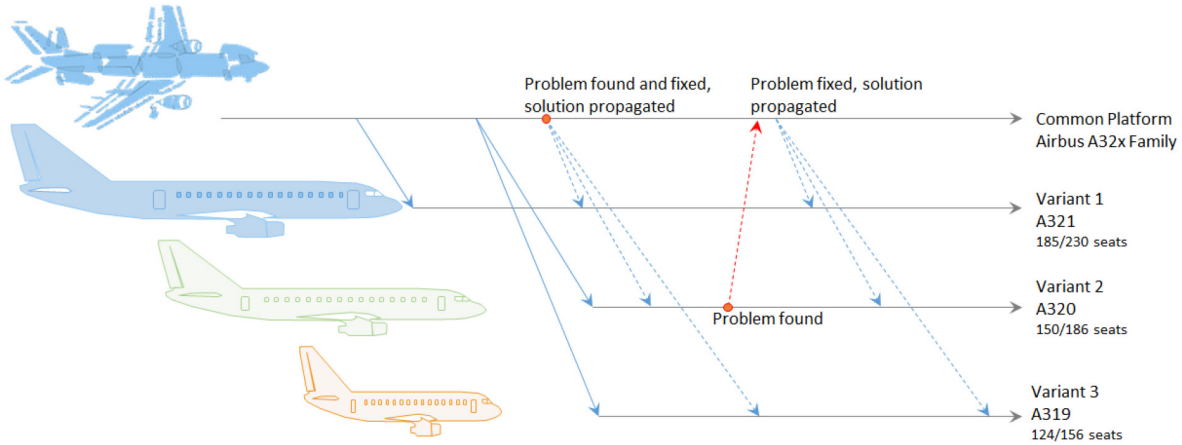
Berger et.al. stellten in den Dagstuhl-Berichten 2019 (Anmerkung 1) fest, dass für die Softwarekonfiguration besonders das Variantenhandling wichtig ist, während das Software Product Line Engineering Unterstützung bei der Versionierung benötigt. Es sei allerdings bisher nicht gelungen, ein übergreifendes Versions- und Variantenmanagement zu etablieren, das sich in der Praxis bewährt. LieberLieber schlägt zur Lösung dieses Problems die Kombination von Modellversionierung und Variantenmanagement mit Hilfe der Werkzeuge Enterprise Architect, LemonTree, Git und pure::variants von PTC vor. Der dargestellte Workflow zeigt eine Lösung unter Verwendung dieser Komponenten:

- Git Feature Branches
- Enterprise Architect Modell
- pure::variants für die Transformation von Varianten

Diese Beschreibung bietet einen ersten Einblick in die gemeinsame Behandlung von Versionen und Varianten eines Modellartefakts. Die Kombination von pure::variants und LemonTree bietet darüber hinaus weiterführende Ansätze (integrierter LemonTree Merge in pure::variants, LemonTree. Components als Varianten, etc.), die wir in einem nächsten Whitepaper näher beleuchten werden.

# VERSIONS- UND VARIANTEN-MANAGEMENT ZUSAMMENFÜHREN

Insgesamt sind Versionierung und Variantenhandling im Engineering komplexer Systeme unverzichtbar, um Qualität, Effizienz und Nachvollziehbarkeit zu gewährleisten und die Herausforderungen von Komplexität und Vielfalt effektiv zu bewältigen.
Während Versionskontrolle für die langfristige Wartung und Weiterentwicklung eines Systems unerlässlich ist, ermöglicht Variantenmanagement die Weiterentwicklung oder Aktualisierung bestehender Versionen eines Systems, ohne die Integrität anderer Varianten zu gefährden.
Wird zum Beispiel auf einer gemeinsamen Plattform eine Produktlinie entwickelt, muss es möglich sein, die unterschiedlichen Ausprägungen des Systems parallel und unabhängig voneinander zu dokumentieren. Wird jedoch auf der Plattform ein allgemeines Problem behoben, ist es schwierig, diese Korrektur konsistent auf die bereits weiterentwickelten Varianten auszurollen.

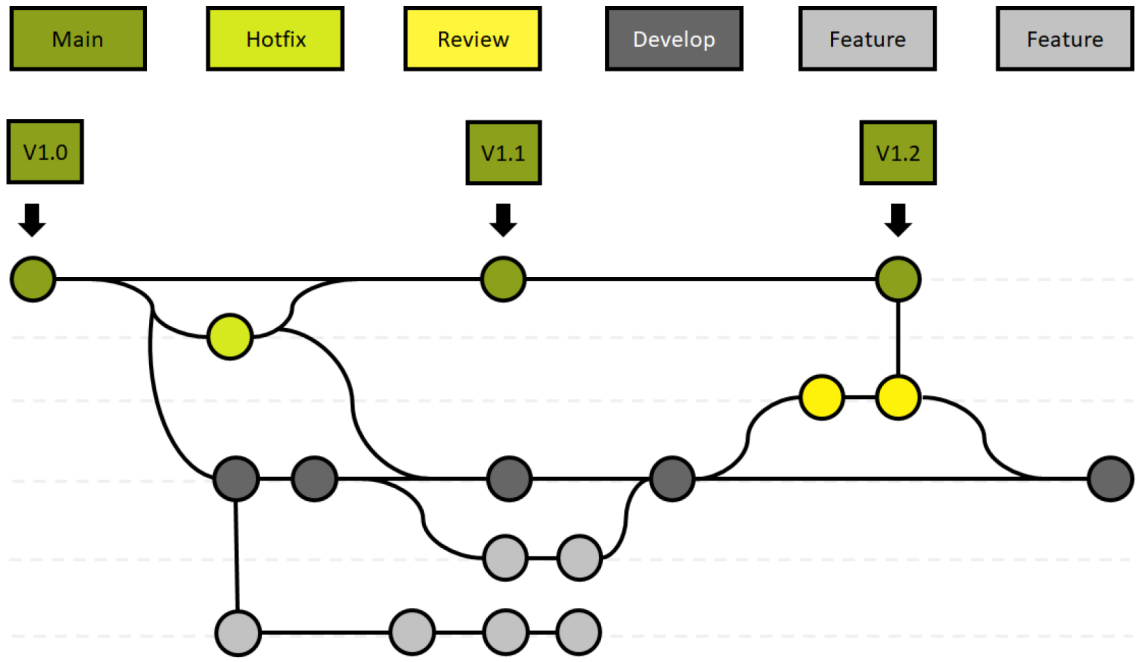



*Variantenmanagement am Beispiel Airbus (Quelle: LieberLieber)*

In der Praxis hat sich gezeigt, dass man entweder ein gutes Versionierungskonzept einführt oder versucht, Varianten zu verwalten. Berger et.al. stellten in den Dagstuhl-Berichten 2019 fest, dass die Softwarekonfigurations-Community regelmäßig mit der Notwendigkeit konfrontiert ist, Varianten zu unterstützen, während das Software Product Line Engineering Unterstützung bei der Versionierung benötigt. Sie meinen, dass es keiner der beiden Communities gelungen ist, einheitliche Techniken für Versions- und Variantenmanagement zu entwickeln, die in der Praxis effektiv sind.

# FEATURE-BRANCH ANSATZ FÜR MBSE

Inspiriert vom Vorgehen aus der Software-Entwicklung wird von LieberLieber seit einigen Jahren bereits eine „Feature-Branch"-basierte Vorgehensweise für MBSE-Artefakte vorangetrieben.




*LieberLieber setzt auf eine „Feature-Branch" basierte Vorgehensweise für MBSE Artefakte (Quelle: https://atyantik.com/coding-smart-with-git-and-gitflow-a-tutorial-for-better-code-management/ )*

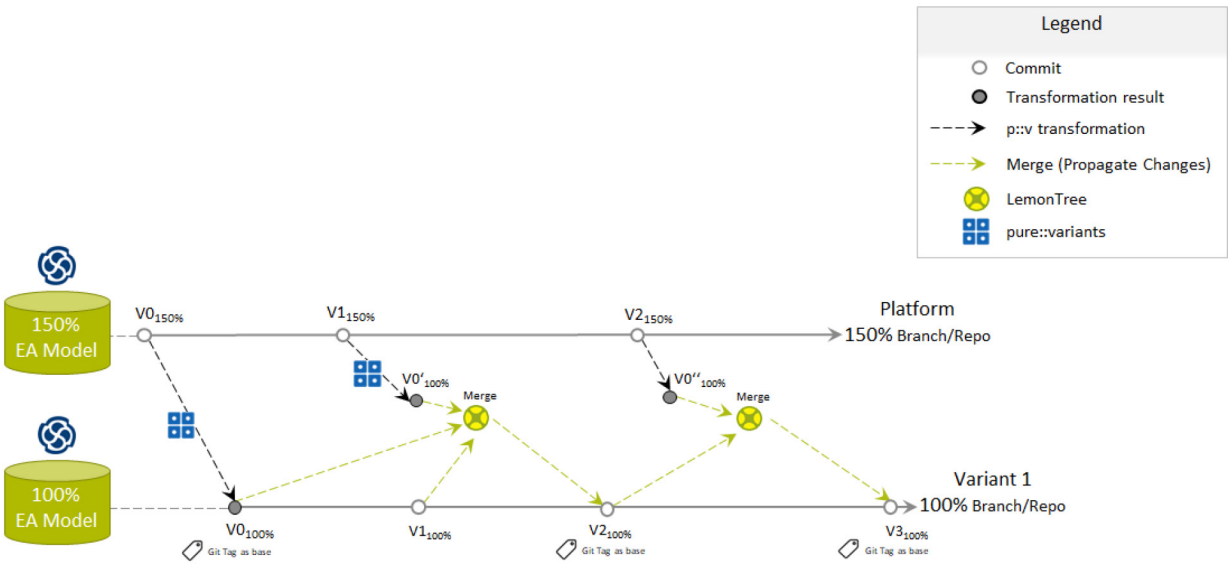
Ein Model-Repository wird dabei ähnlich wie Source-Code behandelt und z.B. mit Git unter Versionskontrolle gestellt. Mit Hilfe von LemonTree ist es möglich, Versionen des Modells zu vergleichen und zusammenzuführen sowie Konflikte zu erkennen und Git Branches zu verwalten (Anmerkung 2). Zum Management der Varianten empfehlen wir pure::variants von PTC, das auf die Definition von Feature-Modellen und die Ausleitung (Transformation) von Varianten spezialisiert ist. Ziel ist es dabei, Feature-Modellierung und Feature-Branch-basiertes MBSE zu verbinden. Auf diesem Weg lässt sich Variantenmanagement bei einem Enterprise Architect Modell anwenden. pure::variants transformiert das komplette Systemmodell in Variantenmodelle, die in weiterer Folge in Git mit Hilfe von LemonTree verwaltet, verglichen und zusammengeführt werden.
Der nächste Abschnitt erklärt, wie ein Systemmodell mit dem 150%-Ansatz mit pure::variants beschrieben und über LemonTree und Git versioniert wird. Dabei lassen sich die Änderungen zwischen Variantenmodellen mit LemonTree und Feature Branches pflegen.

# DAS ZUSAMMENSPIEL VON PURE::VARIANTS, LEMONTREE UND GIT

Zentraler Bestandteil des Workflows ist ein Git Repository, die unterschiedlichen Varianten lassen sich mit Branches abbilden. Das Ausgangsmodell (150%) beinhaltet alle Varianten in Kombination und wird im Haupt-Branch (main) versioniert. Mit pure::variants lassen sich Varianten-Constraints bei Enterprise Architect Elementen, Diagrammen, Konnektoren, etc. hinzufügen. Diese Constraints werden mit dem sogenannten Feature Modell in pure::variants verküpft und definieren, welche Artefakte aus dem Enterprise Architect zu welcher Variante gehören. Durch eine pure::variants Transformation entsteht ein varianten-spezifisches (100%) Enterprise Architect Modell. Das Ergebnis der Transformation wird in einem neuen Branch hinzugefügt und dort mit einem Git-Tag markiert. Dieser Tag dient als Referenz für den Zeitpunkt, an dem die Variante ausgeleitet wurde. Auf diesem Weg wird es möglich, alle ausgeleiteten Varianten-Modelle (100%) sowie das Basis-Modell (150%) unabhängig voneinander und parallel weiterzuentwickeln. Die Verwaltung der Modelle in separaten Git-Branches erlaubt eine unabhängige Entwicklung.
Allerdings muss bei Updates zwischen den Modellen ein sogenannter Merge durchgeführt werden. Beim 3-Wege-Merge mit LemonTree werden drei Modelle für die Zusammenführung berücksichtigt:

- Git Tag der letzten Transformation
- Ergebnis der aktuellen Transformation
- Stand des aktuellen 100%-Modells

Das Ergebnis dieser Zusammenführung (Merge) wird erneut als Git-Tag markiert. Durch einen selektiven Merge mit LemonTree können ebenso Änderungen zwischen verschiedenen 100%-Modellen oder Änderungen von einem 100%-Modell in die Basis (150%) übernommen werden.




*Das Zusammenspiel von pure::variant, LemonTree und Git (Quelle: LieberLieber & PTC)*

# SUMMARY

Durch die Verwaltung der Modelle durch Branches bietet Git uneingeschränkte Möglichkeiten beim Zusammenführen von Änderungen zwischen den verschiedenen Ausprägungen bzw. Varianten eines Modells. Dabei ist das perfekte Zusammenspiel der drei vorgestellten Werkzeuge ausschlaggebend für den Erfolg:

- LemonTree unterstützt bei der inhaltlichen Zusammenführung von Modellelementen
- Git verwaltet die Branches und versioniert das Modell
- pure::variants überblickt das Feature-Modell mit Varianten-Cons traints und führt Modelltransformationen durch

Der oben dargestellte Workflow schildert eine von LieberLieber vorgeschlagene Arbeitsweise, die existierende Tools und Praktiken miteinander kombiniert. pure::variants und LemonTree lassen sich darüber hinaus noch nahtloser integrieren. Ein Beispiel dafür ist etwa die direkte Zusammenführung (Merge) des Enterprise Architect Modells bei der Transformation oder die Verwendung von LemonTree.Automation, LemonTree. Components und pure::variants als Instruktor für die Zusammensetzung von Teilmodellen.

## *Anmerkungen:*

1) Thorsten Berger, Marsha Chechik, Timo Kehrer, and Manuel Wimmer. Software Evolution in Time and Space: Unifying Version and Variability Management (Dagstuhl Seminar 19191). In Dagstuhl Reports, Volume 9, Issue 5, pp. 1-30, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2019)
2) https://customers.lieberlieber.com/downloads/WP_final_DE_Saubere-Versionierung.pdf

*Für den Inhalt verantwortlich:*

LieberLieber

LieberLieber Software GmbH
Gumpendorfer Straße 19
1060 Wien, Österreich
+43 662 90600 2017

**Autor**
Philipp Kalenda
Leitung Consulting

**Bearbeitung**
Rüdiger Maier
Pressearbeit