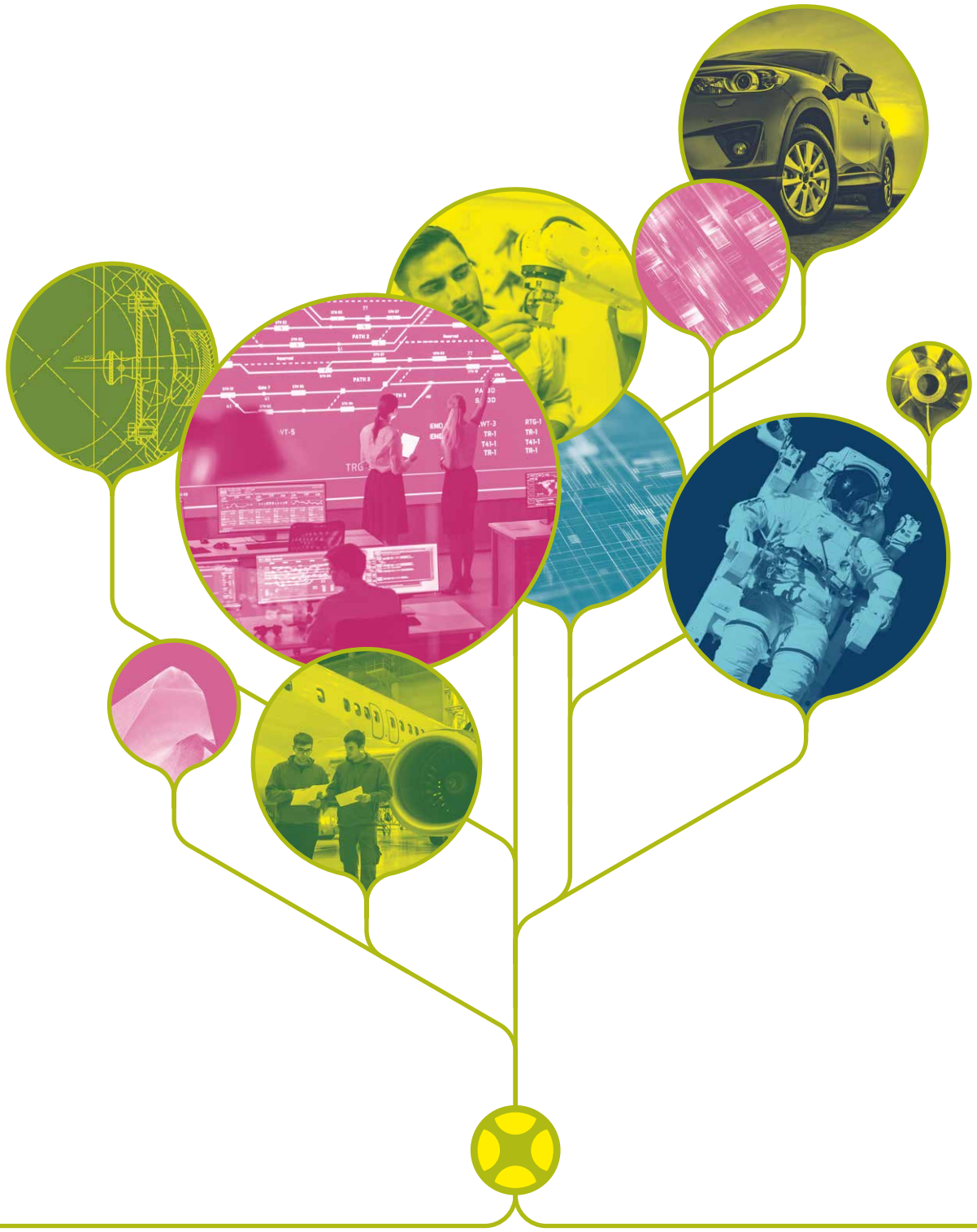


MODELLIERUNG
KOMPLEXER SYSTEME:

IST SAUBERE VERSIONIERUNG ÜBERHAUPT MÖGLICH?

LieberLieber Software

MODELLIERUNG KOMPLEXER SYSTEME: **IST SAUBERE VERSIONIERUNG ÜBERHAUPT MÖGLICH?**

In unserer alltäglichen Praxis des modellbasierten Systems Engineering in verschiedenen Industriebranchen stoßen wir immer wieder auf die Frage nach einer präzisen und einfach handzuhabenden Versionierung von Modellen. Diese Frage wird durch gesetzliche Regulierungen, verschärfte Sicherheitsvorschriften und die starke Zunahme von Produktvarianten immer wichtiger. Daher kommen wir einem vielfach geäußerten Wunsch nach und beantworten in diesem Whitepaper die Frage, ob bei der Modellierung komplexer System überhaupt eine saubere Versionierung möglich ist. Dazu sehen wir uns zunächst an, was ein komplexes System ausmacht und warum bei der Modellierung solcher Systeme eine saubere Versionierung heute schwieriger ist. Dann widmen wir uns den praktischen Dingen, indem wir den Versionierungsprozess näher analysieren und die daraus resultierenden Anforderungen an Werkzeuge beschreiben. In der Zusammenfassung finden Sie schließlich die auf unseren jahrzehntelangen Erfahrungen beruhenden Empfehlungen für die Erreichung einer sauberen Modellversionierung. Folgen Sie uns also auf der Reise in das Land der unbegrenzten Varianten und lernen Sie, wie Sie diese mit Hilfe sauberer Versionierung immer sicher im Griff behalten.

WAS IST EIN **KOMPLEXES SYSTEM?**

Moderne Systeme im Verkehr, Gesundheitswesen etc. sind heute cyber-physische Systeme. Das heißt, dass sie die klassischen mechanischen Komponenten zunehmend mit elektronischen und softwaretechnischen Komponenten verbinden. Ein derartiges System ist durch seinen hohen Grad an Komplexität gekennzeichnet, der gerade auch durch die Ausbreitung von Software schnell weiter anwächst. Wie eine aktuelle Studie (1) zeigt, setzen Unternehmen zur Bewältigung dieser omnipräsenten Komplexität auf die Einführung von Systems Engineering, das inzwischen sogar als erfolgskritisch für die Zukunftsfähigkeit von Unternehmen gesehen wird.

WO WIRD DIE KOMPLEXITÄT IM ALLTAG SICHTBAR?

Auffällig werden die elektronischen und softwaretechnischen Systeme natürlich erst, wenn sie ausfallen - wenn etwa das neue Auto stehen bleibt. Solche Ausfälle verursachen bereits heute einen hohen Anteil der Garantiekosten bei Neuwagen.

Um sich die Dimension der Verbreitung komplexer cyber-physische Systeme vor Augen zu führen, eignet sich der Verkehrsbereich sehr gut. So sind Flugzeuge große Netzwerke aus Bordcomputern und Embedded Systems in vielen einzelnen Aggregaten. Ohne diese Embedded Systems ließen sich nicht einmal die Turbinen starten. Und auch Autos („PC auf Rädern“) und Bahnen sind heute ohne Bordrechner und eingebettete Kontrollsysteme nicht mehr funktionsfähig. Zuverlässigkeit und Sicherheit dieser Systeme sind daher von allergrößter Wichtigkeit. Vernetzte Systeme, die zu Tausenden in einem komplexen Gesamtsystem arbeiten und kommunizieren, setzen eine ganz neue Qualität von Zuverlässigkeit und Sicherheit voraus.

(1) Studie 2021: Systems Engineering in Deutschland - Die deutsche Unternehmenslandschaft im Vergleich – Prozesswerk in Kooperation mit GfSE

WARUM IST EINE SAUBERE VERSIONIERUNG **NOTWENDIG?**

Mit der Verbreitung cyber-physischer Systeme und der rasch zunehmenden Digitalisierung stehen Unternehmen heute vor großen Veränderungen. Dabei treten mechanische und elektrische Anforderungen immer mehr in den Hintergrund, während Software den größten Teil der Wertschöpfung übernimmt. So werden die Systeme aber auch immer komplexer und es treten Herausforderungen auf, die man bisher nicht kannte. Ein zentraler Lösungsansatz dafür ist modellbasiertes Systems Engineering (MBSE). Dabei basieren Informationen über ein zu entwickelndes System nicht mehr auf Dokumenten, sondern auf Modellen. Diese Modelle werden in der Regel auf Basis der UML- oder SysML-Spezifikation erstellt und helfen dabei, zentrale Herausforderungen wie Safety, agile Entwicklungsprozesse und die Arbeit verteilter Entwicklungsteams besser zu bewältigen.

MIT DER VERSIONIERUNG WIRD BEWEISBAR, DASS ALLE SAFETY-VORSCHRIFTEN IM MODELL BERÜCKSICHTIG WURDEN

Gerade in cyber-physischen Systemen gewinnt funktionale Sicherheit immer größere Bedeutung und wird in gesetzlichen Regelungen zwingend vorgeschrieben. Vereinfacht gesagt bezeichnet Safety den Schutz der Umgebung vor einem Objekt. Am Beispiel des Autos kommt es derzeit mit der wachsenden Digitalisierung in Richtung autonomes Fahren zu immer strengeren Vorgaben im Safety-Bereich, damit Autos trotz softwaretechnischer Fahrunterstützung ihrer Umgebung keinen Schaden zufügen. Dabei geht es nicht zuletzt um Haftungsfragen bei einem Unfall oder technischen Versagen.

Da in einem derart agilen und komplexen Umfeld naturgemäß viele Versionen und Varianten von Modellen entstehen, ist eine eindeutige und sichere Versionierung besonders wichtig. Um das Vertrauen zum modellbasierten Systems Engineering zu stärken, muss also klar gezeigt werden, dass eine saubere Versionierung im Entwicklungsprozess jederzeit gewährleistet und auch kontrollierbar ist.

Besondere Bedeutung erlangt die präzise Versionierung durch den Einsatz agiler Methoden, da hier in kurzen Abständen neue Versionen entstehen und man trotzdem den Überblick bewahren muss. Agile Softwareentwicklung bezeichnet Ansätze im Softwareentwicklungsprozess, die die Transparenz und Veränderungsgeschwindigkeit erhöhen und zu einem schnelleren Einsatz des entwickelten Systems führen sollen. Dabei wird die Entwurfsphase verkürzt und darauf geachtet, so früh wie möglich zu ausführbarer Software zu gelangen. Diese wird regelmäßig mit dem Auftraggeber abgestimmt, um die Kundenzufriedenheit zu erhöhen.
Agile Softwareentwicklung zeichnet sich durch selbstorganisierende Teams sowie eine schrittweise Vorgehensweise aus. In internationalen Unternehmen geht die Entwicklung klar hin zu solchen virtuellen Teams, die über regionale, nationale und kulturelle Grenzen sowie Zeitzonen hinweg zusammenarbeiten.

WIE LÄSST SICH EINE **SAUBERE VERSIONIERUNG** VERWIRKLICHEN?

Aufgrund der zuvor beschriebenen Herausforderungen ist es wichtig, eine Versionierungsstrategie zu verfolgen, die jederzeit die Kontrolle über einzelne Versionen und Änderungen garantiert. Zentrales Element ist die Pflege einer klaren Historie über die Entwicklung eines Systems, das durch (parallele) Änderungen entsteht. Dabei müssen einzeln Versionen genau identifiziert und referenziert werden können. So wird es auch möglich, ältere Versionen des zu entwickelten Systems auszuwählen, um diese als Basis für eine Neuentwicklung zu verwenden bzw. eine Variante davon zu erzeugen. Projekte werden nicht mehr von Null auf entwickelt, sondern bauen oft auf bestehenden Entwicklungsleistungen auf. Daher muss man zu jedem beliebigen Punkt in der Entwicklungshistorie zurückgehen können, um von dort an das neue Projekt aufzusetzen.

Ein anderer wichtiger Aspekt ist die persönliche Zuordnung von Entwicklungsleistungen: Welcher Entwickler hat wann welche Änderung durchgeführt. So können bestimmte Änderungen gezielt aus dem Projekt herausgelöst werden, um sie z.B. bei Bedarf rasch rückgängig machen zu können. Sowohl die zunehmende Komplexität als auch immer kürzer werdende Release-Zyklen moderner Softwaresysteme machen es notwendig, verschiedene Modellversionen parallel zu entwickeln. Dazu addiert sich die Herausforderung, dass diese Systeme in immer größeren und verteilten Teams entwickelt werden. Auch im Zuge eines modellbasierten Ansatzes müssen solche Teams die Möglichkeit erhalten, an parallelen Versionen eines Modells effizient arbeiten zu können. Nicht zuletzt ist auch die Einbindung etablierter Versionierungssysteme wie Git in den Modellierungsprozess unabdingbar.

In der klassischen Softwareentwicklung sind agile Entwicklungsprozesse heute weit verbreitet und damit haben sich auch optimistische Versionierungsprozesse durchgesetzt. Auf Basis dieser Prozesse etablierten sich Methoden/Prozesse wie GitFlow, Continuous Integration/Continuous Development oder DevOps. In diesem Whitepaper wollen wir den Prozess GitFlow näher analysieren und wie dieser auch für die Welt der Modellierung verwendet werden kann.

Die zentrale Idee dabei ist es, auf Basis von Feature Branches zu entwickeln. Gitflow ist ein Git-Branching-Modell, das langlebige Feature-Branches und mehrere primäre Branches verwendet. EntwicklerInnen erstellen einen Feature-Branch und verzögern den Merge mit dem Haupt-Trunk-Branch, bis das Feature vollständig ist. Dadurch bringt Git-Flow automatisch eine einheitliche Struktur in jedes Projekt, was vor allem dann von Vorteil ist, wenn viele EntwicklerInnen gleichzeitig an einem Projekt arbeiten. Durch die parallellaufenden Branches können gleichzeitig mehrere Features entwickelt, ein Release vorbereitet und Hotfixes durchgeführt werden. Damit eignet sich das Modell auch hervorragend für größere Projekte.

DIE VORTEILE VON GITFLOW

Der Gitflow-Workflow bietet alle Vorteile des Feature-Branch-Modells: Pull-Requests, isolierte Experimente und eine effizientere Zusammenarbeit. Der Workflow nutzt ebenfalls ein zentrales Repository als Kommunikations-Drehkreuz für alle EntwicklerInnen.

ANFORDERUNGEN AN WERKZEUGE

Für die praktische Umsetzung einer sauberen Versionierung bei der Entwicklung komplexer Systeme bedarf es geeigneter Werkzeuge. In diesem Absatz zeigen wir auf, welche Anforderungen an diese Werkzeuge sich aus den oben geschilderten Umständen ergeben.

PRÄZISE ERKENNUNG UND ZUSAMMENFÜHRUNG VON ÄNDERUNGEN

In der Entwicklungsphase komplexer Systeme entstehen naturgemäß unzählige Modellversionen. Daher ist die wichtigste Funktion eines Werkzeugs der detaillierte Vergleich und die einfache Zusammenführung verschiedener Versionen. Das bedingt eine präzise und fein-granulare Erkennung von durchgeführten Änderungen. Herkömmliche Ansätze in der Softwareentwicklung verwenden dafür zeilen- und text-basierte Anwendungen, die jedoch bei grafischen Modellen nicht ausreichend sind. Zusätzlich ist auch das Wissen über die Semantik der Modellierungssprache wichtig. Ein Änderungsbericht auf Datenbankebene oder durch XML-Dateien ist weder nutzerfreundlich, noch ermöglicht er eine saubere Zusammenführung von Änderungen.

BENUTZERFREUNDLICHE DARSTELLUNG VON ÄNDERUNGEN

Um mit einem Werkzeug gut versionieren zu können, ist nicht nur die präzise und vollständige Änderung auf Modellebene wichtig, sondern auch die leicht verständliche Darstellung der Änderungen. Modellelemente können nämlich verschoben, geändert, gelöscht oder neu hinzugefügt werden. Darüber hinaus werden solche Elemente oft in mehreren Diagrammen dargestellt. Um dabei den Überblick zu behalten, müssen die Änderungen einfach erkennbar sein. Nur so ist es dem Entwickler möglich, sich ein gutes Bild über das Geschehen zu machen und auch die Änderungen anderer Mitwirkender verstehen zu können.

KONFLIKTE BEI ÄNDERUNGEN UND IHRE WECHSELSEITIGEN ABHÄNGIGKEITEN

Wenn mehrere Personen in Modellen Änderungen durchführen, können dadurch Konflikte auftreten. Daher muss ein Werkzeug diese Konflikte präzise kalkulieren. Ein Konflikt soll nämlich erst dann gemeldet werden, wenn dieser nicht automatisch bei der Zusammenführung aufgelöst wird und eine menschliche Interaktion notwendig ist. Dann müssen sich mit Hilfe des Werkzeugs diese Änderungen einfach zusammenführen und so die Konflikte beseitigen lassen. Bei diesem Prozess ist es Aufgabe des Werkzeugs, die Validität des Modells verlässlich sicherzustellen.

Da in der Praxis bereits bewährte Prozesse, Workflows und Werkzeuge eingesetzt werden, müssen Modelle damit interagieren können. Das Modellierungswerkzeug muss daher eine nahtlose Integration in bestehende Versionierungssysteme (SVN, Git, PTC etc.) und industrielle Application-Lifecycle-Management-Werkzeuge (ALM) ermöglichen.

BEARBEITUNG VON TEIL-MODELLEN

In den Entwicklungsprozess komplexer Systeme sind heute ganze Lieferantenketten einbezogen, die jeweils nur einen Teil zum Gesamtsystem beitragen. In der klassischen „Produktlinienentwicklung“ ist es daher unumgänglich, dass mit einem Werkzeug auch Teile von Modellen durch verschiedene Zulieferer bearbeitet und anschließend wieder in das Gesamtsystem integriert werden können. Erschwert wird dieser Prozess noch durch unzählige Produktvarianten und den Trend zu sogenannten „Plattform-Strategien“ (z.B. eine Bodengruppe für verschieden Fahrzeugtypen).
Damit muss gewährleistet sein, dass verschiedene Elemente und Teile des Gesamtsystems von verschiedenen Entwicklerteams zu unterschiedlichen Zeiten und an unterschiedlichen Orten bearbeitet und letztlich wieder in das Gesamtsystem integriert werden können. So ist ein Gesamtsystem eigentlich ein System von vielen Subsystemen. In diesem Szenario muss ein Werkzeug sicherstellen, dass es weder bei der Herauslösung noch bei der Reintegration von Modellteilen zu Systemkonflikten kommt bzw. Informationen verloren gehen.

HERMANN GOLLWITZER, BEI VW ALS SYSTEM-ARCHITEKT FÜR INFOTAINMENT SYSTEME TÄTIG:

„Mit tatkräftiger Unterstützung durch LieberLieber ist es uns gelungen, modellbasiertes Systems Engineering (MBSE) zum Rückgrat unserer Entwicklungsorganisation zu machen. Die Kontrolle aller Versionen und Varianten der im Entwicklungsprozess entstehenden Modelle war für uns die wichtigste Anforderung zur stabilen Verankerung von MBSE im Unternehmen. Dabei hat uns LieberLieber mit profundem Praxiswissen sowie mit Werkzeugen wie Enterprise Architect und LemonTree sehr gut unterstützt und damit die interne Überzeugungsarbeit gefördert.“

ZUSAMMENFASSUNG UND **EMPFEHLUNGEN**

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 haben wir uns in diesem Whitepaper mit der Frage befasst, ob eine saubere Versionierung in komplexen Systemen überhaupt möglich ist.

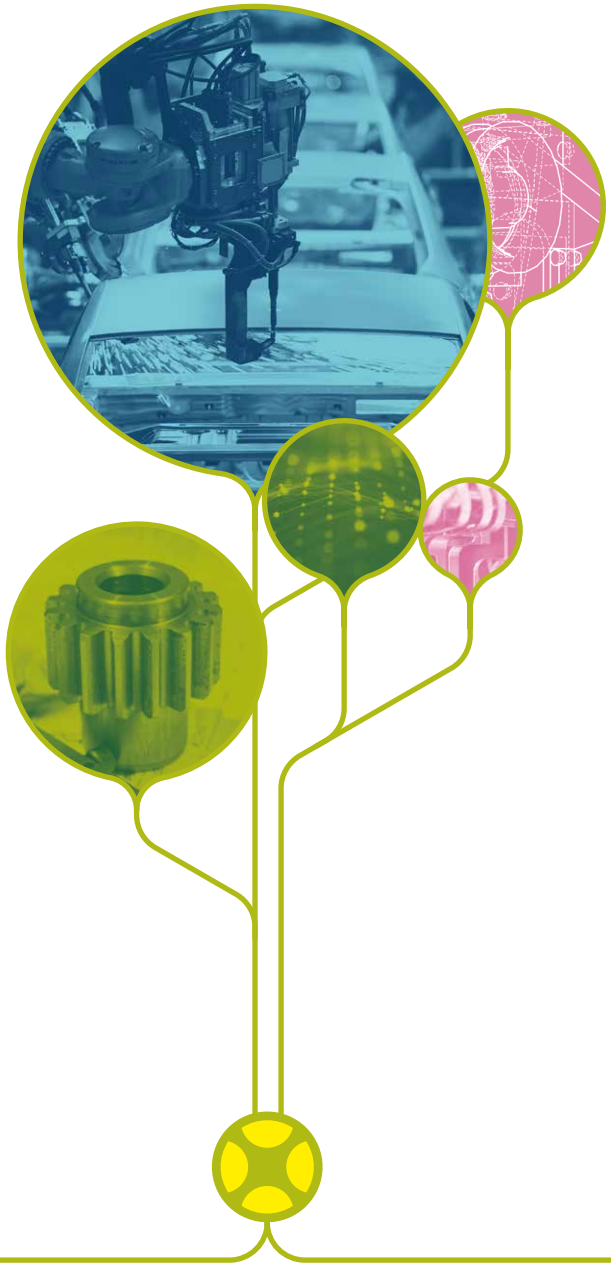
Unsere Antwort darauf laut ganz klar Ja, allerdings bedarf es dazu entsprechender Erfahrungen und geeigneter Werkzeuge. Unserer jahrzehntelangen Erfahrung zufolge befinden sich in der von uns zur Umsetzung empfohlenen Werkzeug-Kette rund um Enterprise Architect, LemonTree und Git aktuell die besten Voraussetzungen, um eine saubere Versionierung am letzten Stand der Technik zu verwirklichen. Unserer Einschätzung nach ist das Potential dieser Werkzeug-Kette am Markt derzeit einzigartig und eröffnet unseren Kunden ganz neue Möglichkeiten.

LEMONTREE:
DAS RICHTIGE WERKZEUG FÜR DIE SAUBERE VERSIONIERUNG

LemonTree wurde speziell dafür entwickelt, die oben angeführten Anforderungen an Werkzeuge zu erfüllen:

- Präzise Erkennung und Zusammenführung von Änderungen
- Benutzerfreundliche Darstellung von Änderungen
- Präziser Umgang mit Konflikten bei Änderungen und ihren wechselseitigen Abhängigkeiten
- Bearbeitung von Teil-Modellen
- Wiederverwendung bewährter Prozesse (Git, DevOps etc.)

Die für LemonTree immer schon zentrale Aufgabe, die teambasierte Zusammenarbeit in den heute üblichen komplexen Projekten bestmöglich zu unterstützen, wird mit neuen Releases immer weiter ausgebaut. So lässt sich ein Teilmodell (oder das Gesamtmodell) weiterhin als EAP(x) in Versionierungssystemen wie Git verwalten, was die Arbeit der verteilten Teams wirkungsvoll unterstützt. Wenn sich eine Komponente des Modells während der parallelen Bearbeitung entwickelt hat, kann sie dank der intelligenten Merge-Funktion leicht wieder importiert werden. Die in LemonTree integrierte Abhängigkeitsanalyse dient der genaueren Definition von Modellteilen vor dem Export. Wechselseitige Abhängigkeiten lassen sich damit begutachten oder bei Bedarf beseitigen. Einfache Abhängigkeiten und zyklische Abhängigkeiten zwischen Modellpaketen werden klar aufgezeigt.



Für den Inhalt verantwortlich

LieberLieber

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

Die Autoren

Dipl.-Ing. Rüdiger Maier, M.A.
Leitung PR

Dr. Konrad Wieland
Geschäftsführer

Kontaktieren Sie uns bitte unter
welcome@lieberlieber.com