LEMONTREE AUTOMATION

How to work **successfully** with
LemonTree Components in a
build pipeline using version control

Whitepaper

TABLE OF CONTENTS

INTRODUCTION

Lifecycle management is an important topic in the development of cyber-physical systems. These systems are specified, developed and verified using model-based approaches. Often, an overall model is created that includes several subsystems or components. If the overall model is placed under version control, all components must be managed together over their life cycle (releases, variants, etc.).

In order to operate lifecycle management for parts of the model or individual components, the model must be split up. The resulting sub-models can be maintained individually. LemonTree Components makes it possible to specify parts of an Enterprise Architect model as reusable components and to separate them from the model. These components can be edited as an independent model, versioned and fed back into the overall model.

Depending on the working method, the overall model is regularly updated and a new revision of the components is imported. This task is taken over by LemonTree Automation, integrated in a build server pipeline. In this way, models are integrated into familiar processes and changes remain visible and testable.

This white paper describes how LemonTree Components and LemonTree Automation are integrated into a DevOps process to efficiently manage the lifecycle of partial models.

LEMONTREE **COMPONENTS**

LemonTree Components allows the division of a model created with Enterprise Architect into different sub-models or components. These can be easily distributed, reused and, if required, integrated back into the overall model.
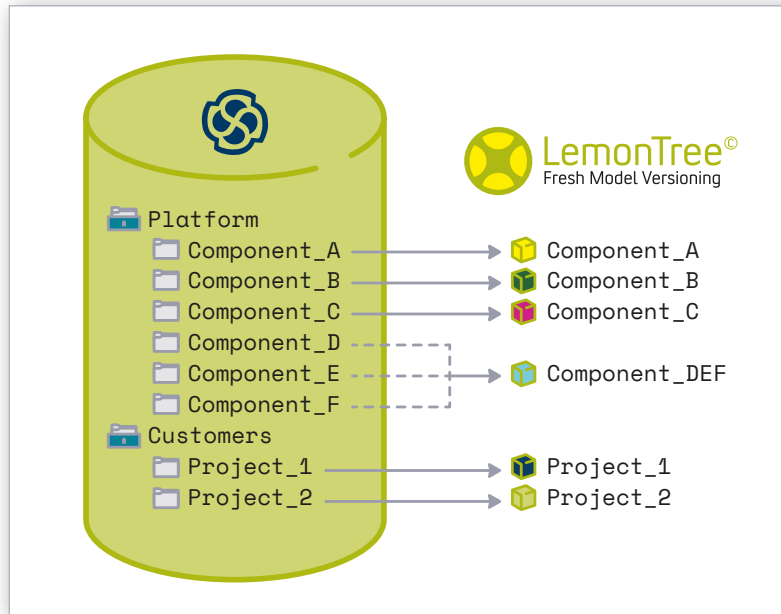

Figure 1: From overall model to model components

LemonTree components are extracted from the model by export and imported into another model by import. This restricts the distributed editing to specific parts of the model. With LemonTree, differences in the edited components can be recognised and visualised. After adjusting the component in a so-called working model, the changes can be imported back into the original overall model. If the content of the components was changed on both sides (in the overall model as well as in the working model), LemonTree enables the creation of a new, merged version of the component. If there is a conflict between the changes, LemonTree assists in resolution and consolidation.

LEMONTREE **AUTOMATION**

The above mentioned function of LemonTree Components can be executed with the LemonTree Addin as well as via the command line interface of LemonTree Automation.

LemonTree supports the following features:

▶ - Export of LemonTree Components

▶ - Import of LemonTree Components

▶ - Merge of Enterprise Architect models (only without conflicts; in case of a conflict, LemonTree terminates with an ExitCode)

▶ - Compare Enterprise Architect models

▶ - Create „Single File Sessions" (file that contains the diff information and can reproduce the used LemonTree Session)

LemonTree Automation is primarily used in the context of a build server pipeline to update Lemon-Tree Components (based on commits) or to create Diff Sessions that can be used as review artefacts.

The next chapters describe a possible workflow to integrate LemonTree Components together with LemonTree Automation in a build server pipeline.

EXAMPLE WORKFLOW:
## INTEGRATION INTO A
BUILD PIPELINE

The following workflow shows one way to work with LemonTree Components and Automation in the context of a build pipeline. By exploiting all the possibilities of the tools, an extensive tool chain is created. An overview of the different expansion stages is given in the chapter „Different expansion stages of the example workflow".

In principle, Enterprise Architect models can be managed in version control systems and thus used in a build pipeline. However, one problem with this is that under certain circumstances a monolithic overall model is versioned that consists of various, independent components. To solve this problem, the model is split into LemonTree Components and each component is managed in its own version control repository (called a Git repository in this example).

The first step in this process is the specification of LemonTree Components. This defines which Enterprise Architect packages are combined into a logical component.

SPECIFICATION OF LEMONTREE COMPONENTS

The specification of a LemonTree Component is created via the LemonTree Addin. The packages that are to become part of the LemonTree Component are selected. A component can optionally be marked as „Read-Only", which makes it uneditable after import into a working model. Thus, the component is usable but cannot be modified.
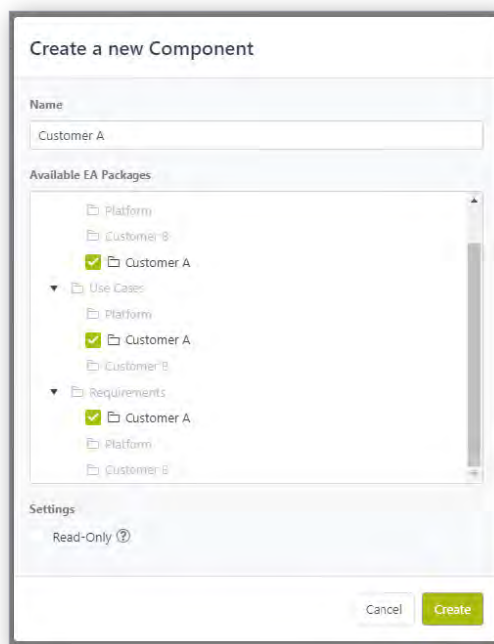
When exporting a component from the model, its dependencies must be taken into account. These references are recognised and displayed by LemonTree. Important are the outgoing references that represent a dependency.

Figure 2: Specification of a LemonTree Component consisting of several EA packages

DEPENDENCIES OF LEMONTREE COMPONENTS

Dependencies between components result from references in the model, such as classifier references, relations or graphical representations.
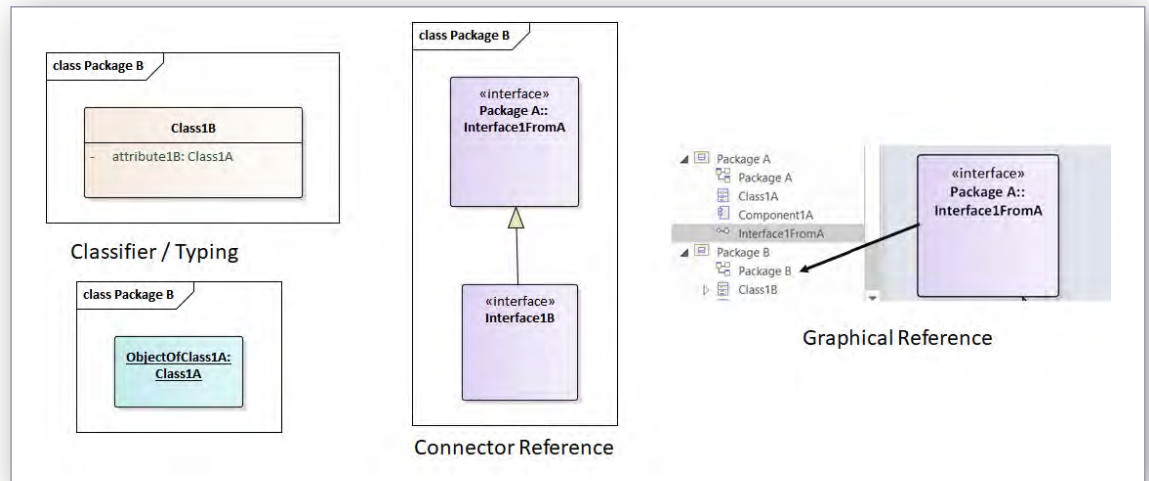

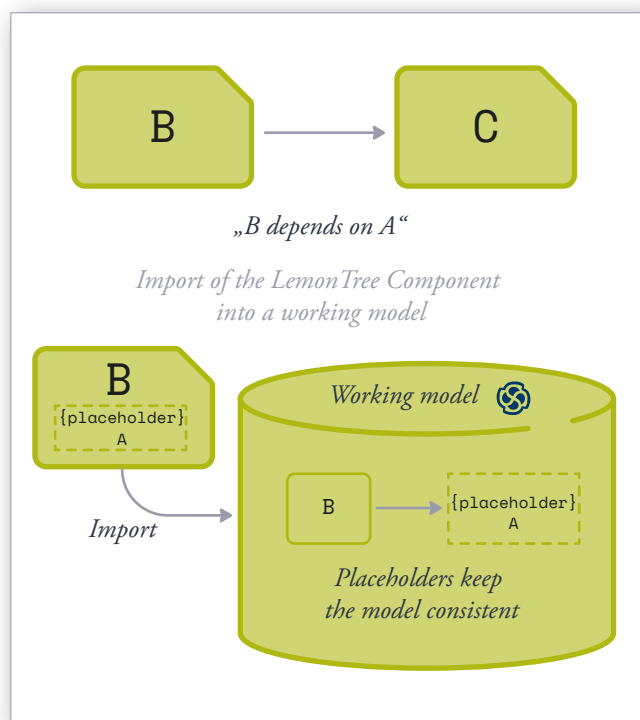*Figure 3: Types of references for model elements*

Figure 4: Import of a LemonTree component with dependencies

In a model, the source and target of a reference are always present. If a model is cut into different components, the source and target of a reference may be separated. This would cause the reference to be lost, as only one side of the reference is available in the split model. Therefore, LemonTree Components work with so-called placeholders: They act as substitute elements to resolve the missing end of a reference.

Through the mechanism of placeholders, a consistent model is always created for any constellation of components. However, certain dependencies may not be desired due to modelling guidelines or principles, such as circular dependencies.

RESOLVING DEPENDENCIES

If a certain constellation of dependencies is not desired, it is resolved with the dependency overview.
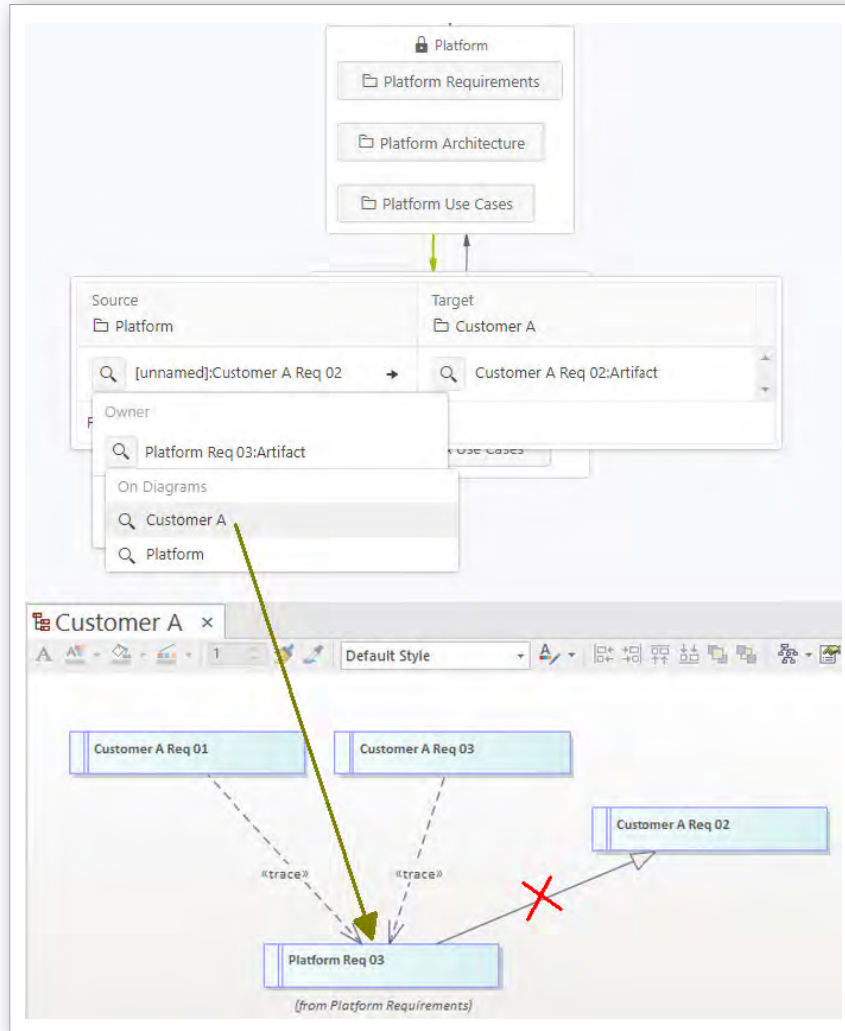

Figure 5: Resolving a circular dependency between LemonTree Components

In the example in Figure 5, an undesirable circular dependency between the platform component and a project component is resolved. Behind the circular dependency is an erroneously created generalisation relationship between a platform and a customer requirement. LemonTree allows you to navigate to the origin of references and then resolve them in Enterprise Architect. In Figure 5, this relates to the deletion of the generalisation relationship.
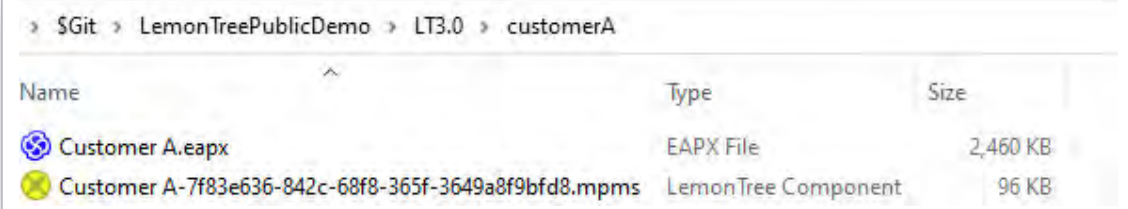
Once all LemonTree Components have been specified and unwanted dependencies resolved, the components are exported and stored in separate Git repositories.

WORKING WITH LEMONTREE COMPONENTS IN VERSION CONTROL

When exporting a LemonTree Component, a MPMS (Model Package Management System) file is created. This file is based on JSON and is checked into the Git repository and managed there. Since the MPMS file is only an extract of the component, a working model is created for editing the component with LemonTree Automation. Unlike LemonTree Component, this is not managed in version control.

CREATION OF A TEMPORARY WORKING MODEL

To process a LemonTree component, a temporary working model is created into which the component (with all dependencies) is imported using LemonTree Automation. The flow logic for the creation of the working model as well as the provision of the referenced dependencies is handled by a script (e.g. Powershell or Python). The script copies an empty Enterprise Architect model (as a template) and imports the LemonTree component to be processed into it.

Figure 6: Working model for the LemonTree component „Customer A

In addition to the primary LemonTree Component, all components to which an outgoing reference exists are also imported. The aim is to create a working model without placeholders. The referenced elements are imported as read-only components to prevent them from being changed in this working model.
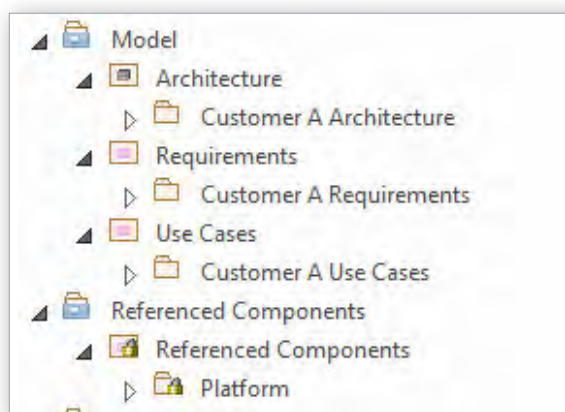

Figure 7: Imported component „Customer A" with read-only component „Platform

The adjustment of a referenced component must be done in the corresponding Git repository. Subsequently, an update of the referenced component is imported.

Since the Enterprise Architect model itself is only temporary and is not managed in version control, the LemonTree component must be exported back into the MPMS format to finally publish it with a Git commit and push.

EXPORT OF THE MODIFIED LEMONTREE COMPONENT

To publish the changes of a LemonTree Component, the content of the working model is exported. For this purpose (as with the import) a script is used that controls LemonTree Automation. The MPMS file in the Git repository is overwritten with the export of the LemonTree component from the working model. The changes are then published via Git Commit and Push.
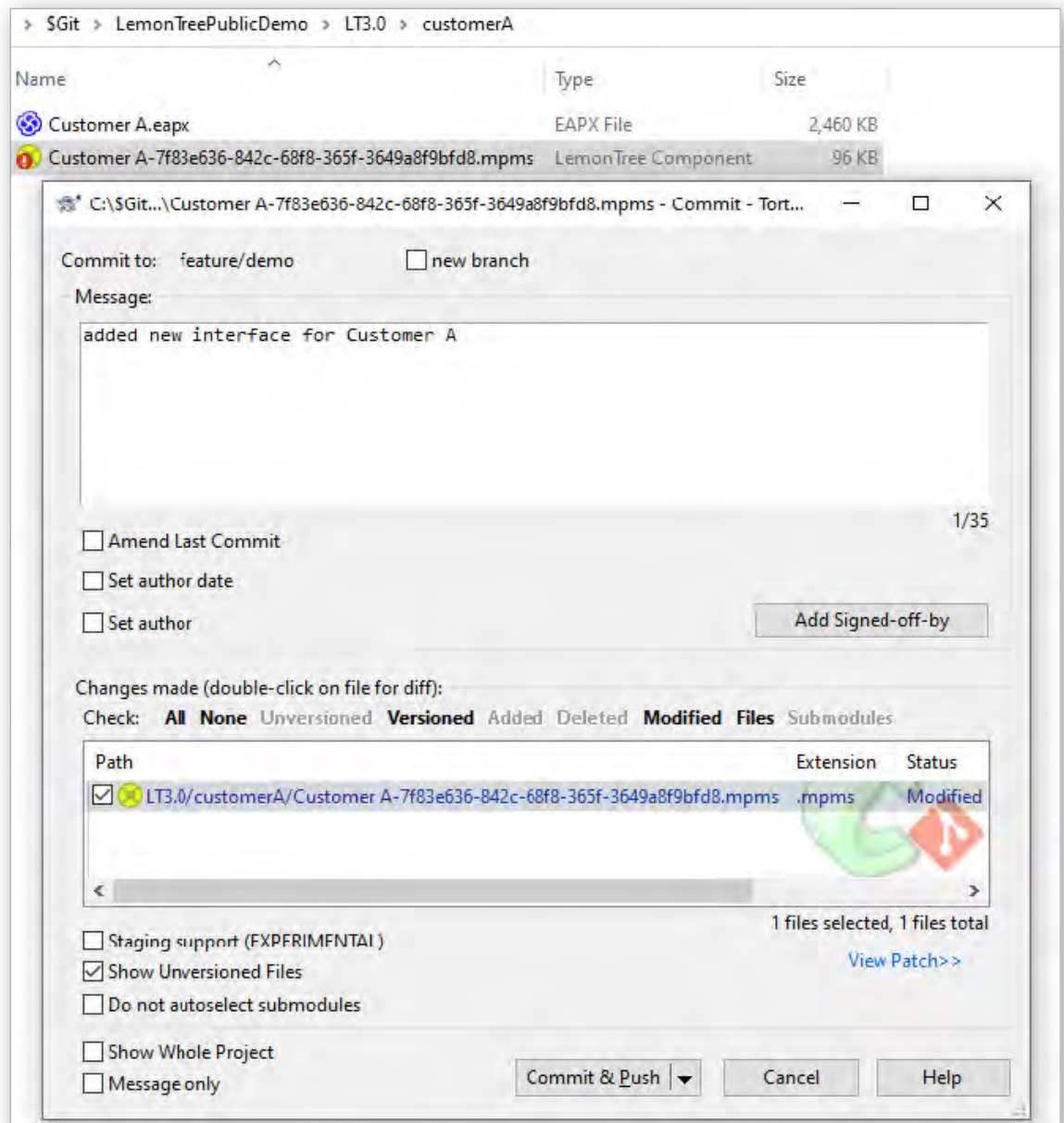

Figure 8: Release of a modified LemonTree Component in Git

Publishing the change of a LemonTree Component via Git Push on the Git Server triggers a series of automated actions on a build server.

BUILD-SERVER PIPELINE

In this document, the term „build server pipeline" is used as a proxy for various build server systems and build processes - e.g. Azure DevOps, TeamCity, Jenkins, GitHub Actions. In such a pipeline, certain actions are defined that are executed in the context of a version control repository as soon as a new revision is published in the Git repository (via commit & push). Depending on the files under version control, different actions are executed.
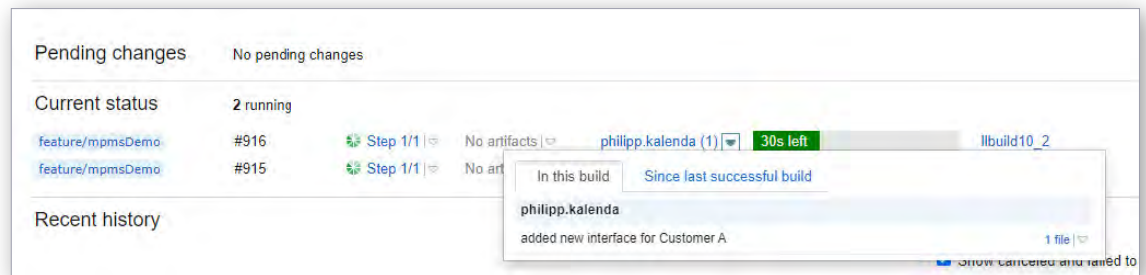

Figure 9: Build run after committing the MPMS file

In software development, for example, projects are compiled, tested and verified. In the case of versioned models, validations can also run (e.g. using SQL queries in the Enterprise Architect model) or a comparison of certain versions is carried out with the help of LemonTree.

Diff and merge operations are suitable for versioning LemonTree components.

AUTOMATED COMPARISON OF LEMONTREE COMPONENTS

To easily track each change, each triggered build run calculates the difference between the current version and the previous version. Since the trigger of a build run is always the current commit, the LemonTree Component (MPMS file) of the current commit is available in this context. In addition, the build script fetches the previous version from the Git repository and starts a comparison with LemonTree Automation. Since LemonTree Automation runs during the build process and does not provide a user interface, LemonTree Automation creates a so-called „single file session". This artefact contains the compared models or LemonTree components so that their comparison can be reproduced at any time. This session artefact is published as a result at the end of the build run and can thus be viewed by anyone with access to the build project.
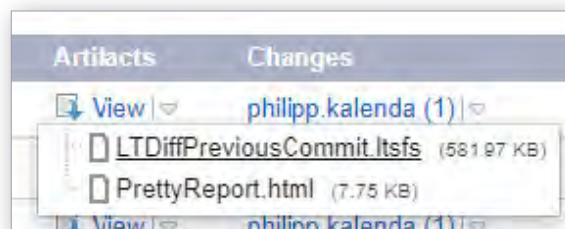
The LemonTree Single File Session can be downloaded and opened to see the comparison with the last commit of the LemonTree Component.

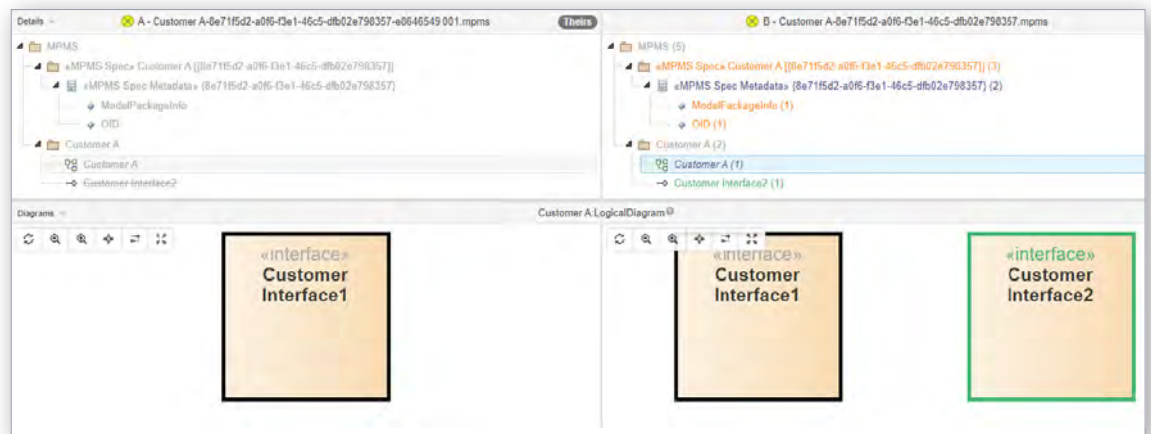*Figure 10: Generated diff session as build artefact*

Figure 11: Diff of a LemonTree Component from LemonTree Single File Session

In addition to automated comparisons, the calculation of possible conflicts in merge operations is also supported. This is especially necessary when working with a multi-branch strategy (e.g. with GitFlow).

MERGE PREVIEW OF LEMONTREE COMPONENTS (MULTI-BRANCH STRATEGY)

When working with a multi-branch strategy (e.g. GitFlow), created branches are merged again. This happens, for example, when a feature is completed or a release is published. However, before a change is adopted or integrated into a branch, a merge or pull request is first created. This determines which change is merged into which branch. In order to assess whether a change can be adopted or released, a review of the changes from the branch is carried out. In order to better understand the model changes, a LemonTree Single File Session is also created in this case. A comparison between the merge target (e.g. the develop branch) and the current branch (e.g. feature branch) is calculated.
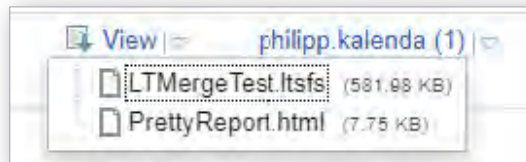

Figure 12: Merge preview artefact

This merge artefact shows in a LemonTree session what the final result of the LemonTree component would look like after the merge. If a conflict occurs in the context of the merge process, it cannot be resolved automatically. In this case, a diff session is created and the build process terminates with an error message.

Figure 13: Failed build due to conflict

If, for example, a feature is completed and a branch is merged, a new version of the component exists after the rebuild process. In order to integrate this component in the overall context, an overall model is regularly created or updated in which all existing LemonTree components are merged.

INTEGRATION INTO AN OVERALL MODEL

The overall model is usually managed at a central location - such as a file share or a database server. It represents the entirety of all components in certain versions. In order to regularly import updates, the overall model is updated depending on the build processes of the LemonTree components.

If, for example, a pull request of a component is completed, the changes from the feature branch are merged back to develop. This merge triggers a new build in the develop branch, which updates the overall model and imports the latest version of the corresponding component. The result of the build is a LemonTree session for the comparison between the current develop commit and its predecessor as well as an overall model with the latest state of the LemonTree component.

The overall model only serves as a source of information or as a reference to check the compatibility of the components with each other. The changes to the components are always made in the respective Git repository, using LemonTree Component, LemonTree Automation and the generated working model.

DIFFERENT EXPANSION STAGES
OF THE EXAMPLE WORKFLOW

The workflow described above gives an example of how to work with LemonTree Components and Automation. It shows the current maximum expansion level of LemonTree tools in the context of a build pipeline. However, it is not absolutely necessary to set up this comprehensive form of the tool chain. Therefore, we now describe the different expansion stages of the example workflow. This makes it clear that this way of working can be introduced step by step or according to your requirements.

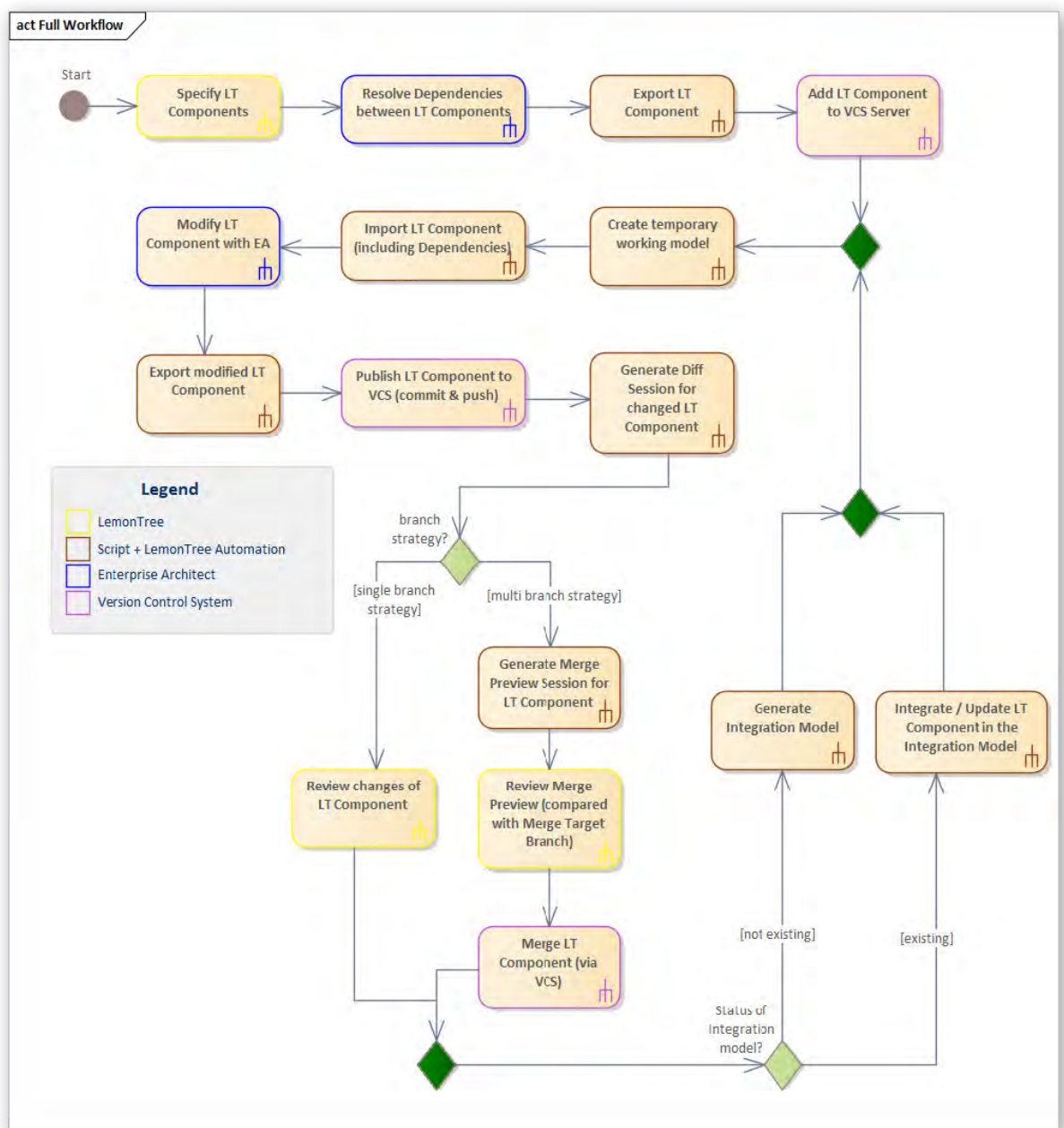The entire sample workflow looks like this:

Figure 14: Overall example workflow

VARIANT WITHOUT LEMONTREE COMPONENTS
(entire Enterprise Architect model is versioned)

This variant does not work with LemonTree Components, but with an entire Enterprise Architect model (eap, eapx or qeax file). It is suitable for small and medium-sized models and also if there are no components in a model that are to be managed in their own life cycle.

With this variant, the steps „Specification of LemonTree Components", „Working with Lemon-Tree Components in Version Control" and „Integration into an Overall Model" are omitted. The Diff Sessions and Merge Previews - described in the chapter „Build Server Pipeline" - can also be created with a complete Enterprise Architect model on the Build Server.
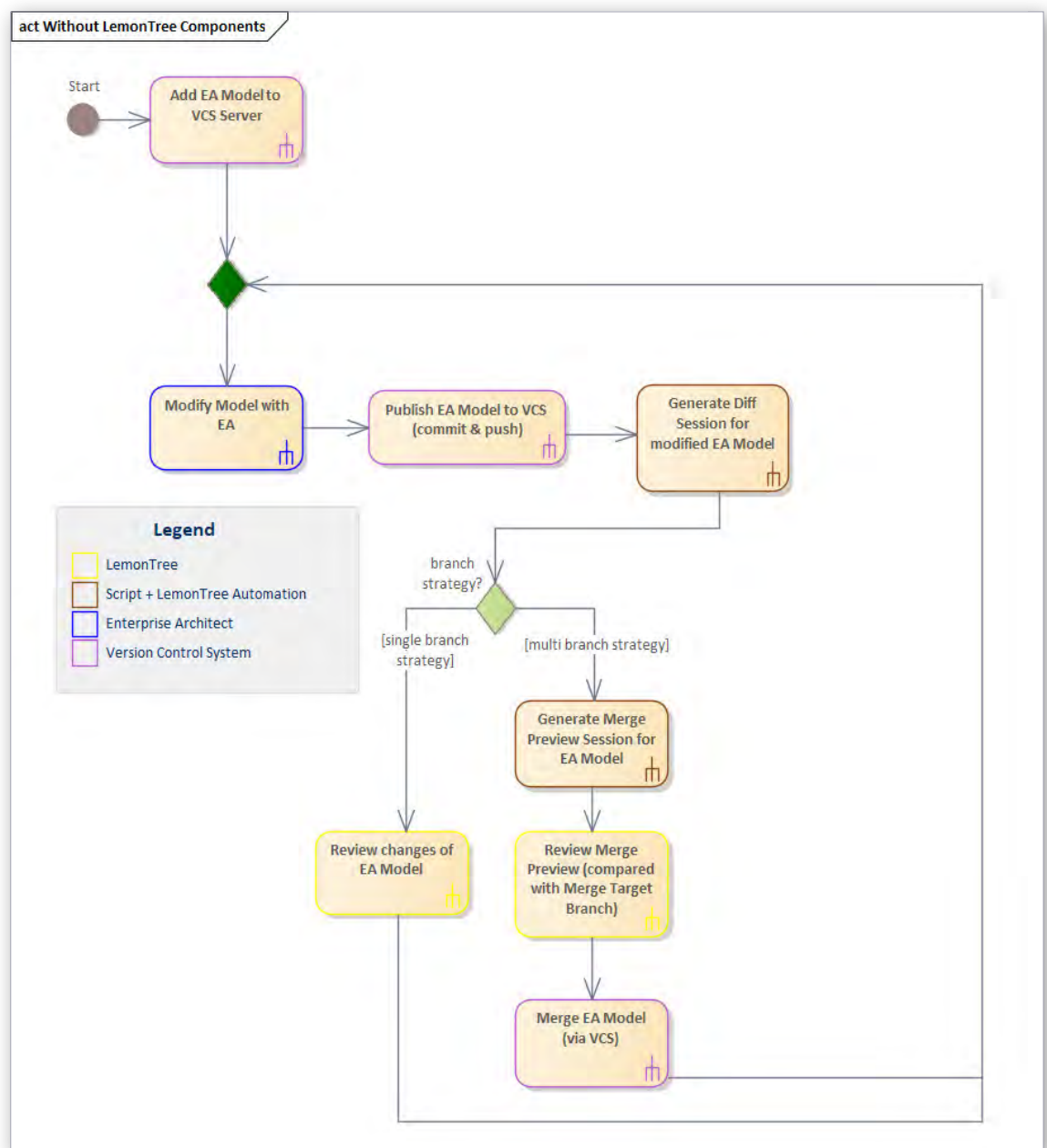

Figure 15: Workflow with EA Model under version control

VARIANTS WITH LEMONTREE AUTOMATION IN DIFFERENT STAGES OF EXPANSION

LEVEL 1: WITHOUT LEMONTREE AUTOMATION

The described workflow is also possible without using LemonTree Automation. However, both the exchange of LemonTree Components and the comparison of changes become manual activities. No build server is used and the update of the integration model with all LemonTree Components is done manually via the LemonTree Addin.
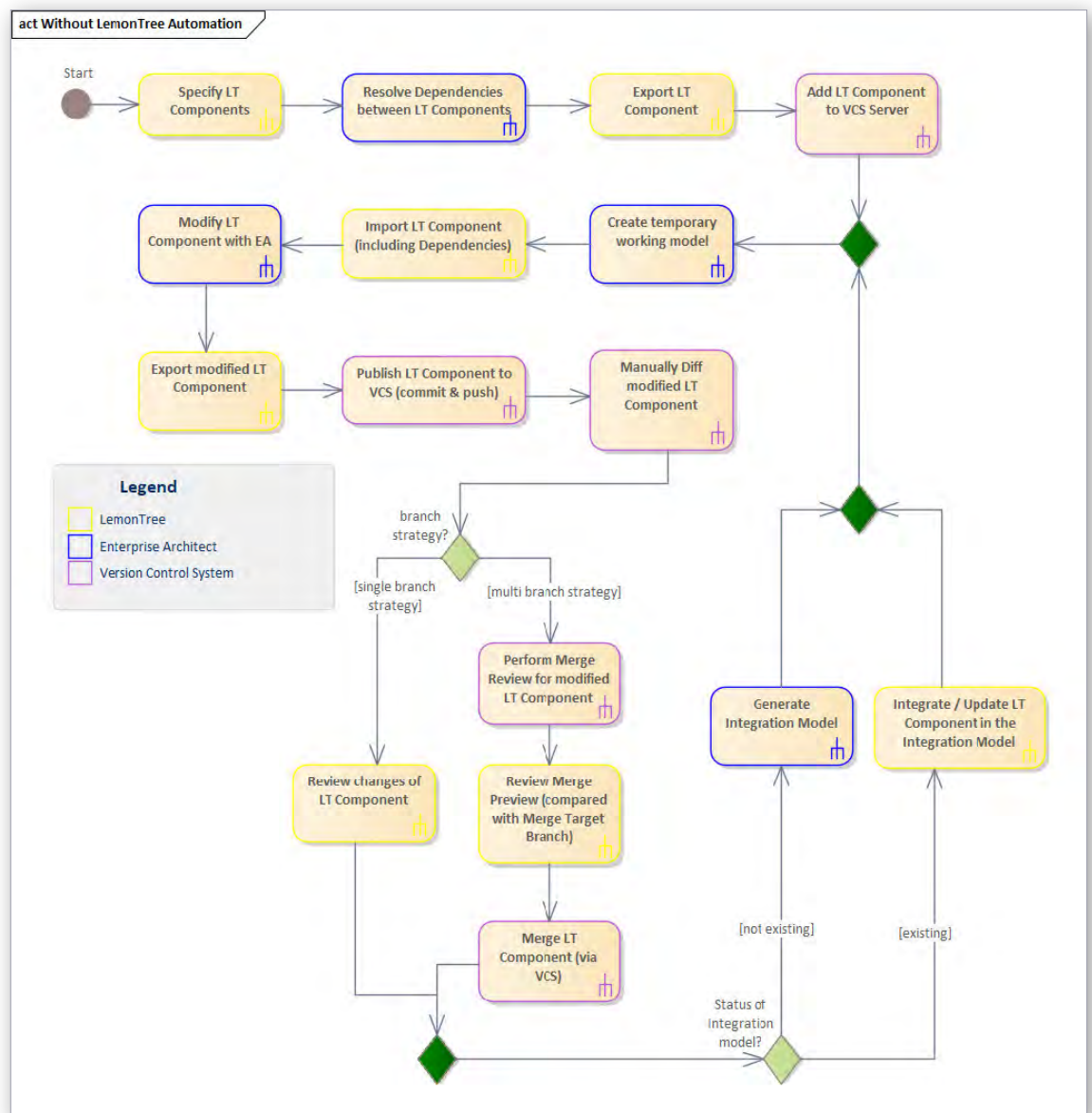

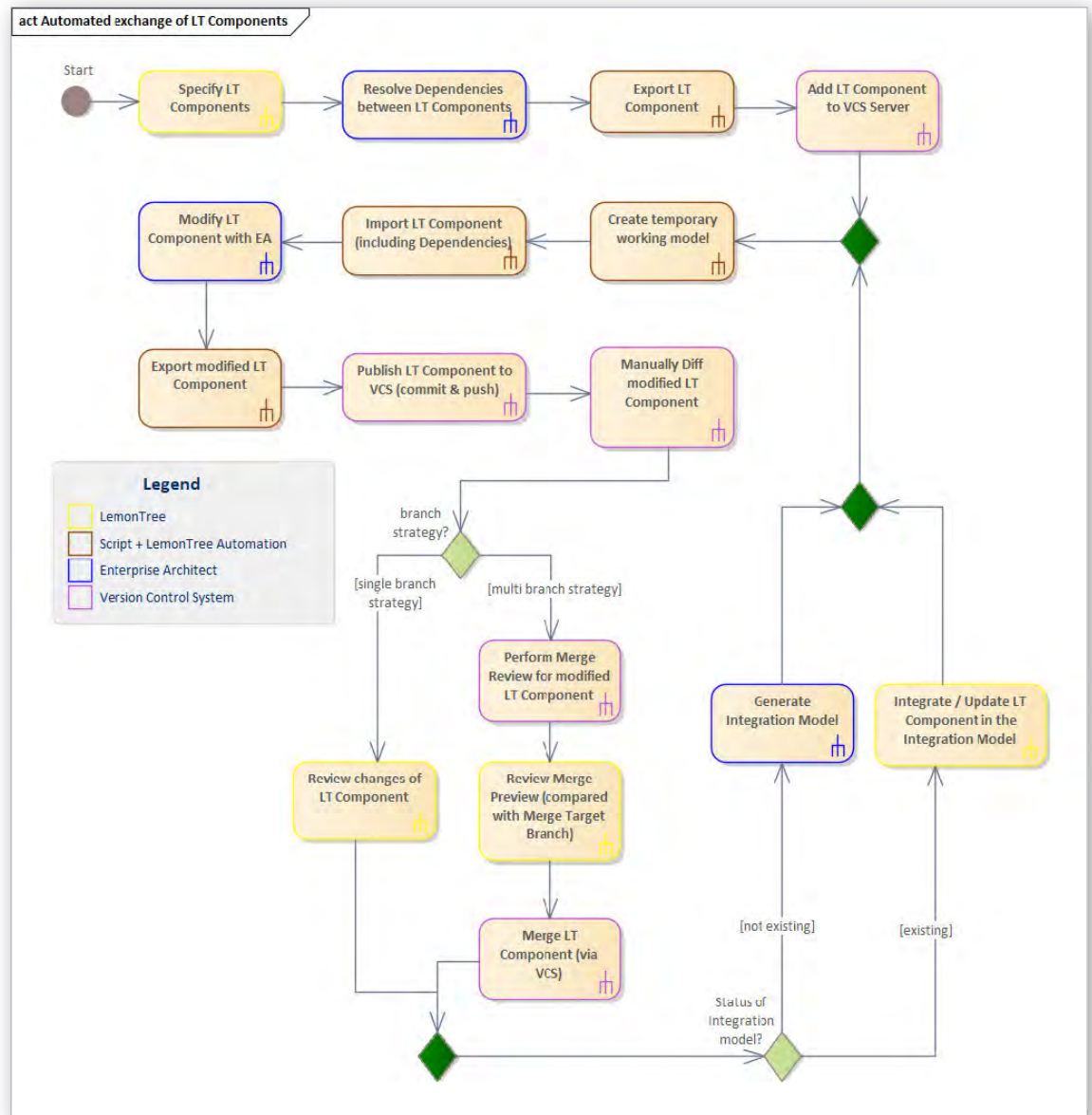*Figure 16: Manual workflow without LemonTree Automation*

LEVEL 3: DIFF / MERGE REPORT ON THE BUILD SERVER

In stage 3, a build server is also used to create automated diff/merge reports. The update of the integration model with all LemonTree Components is done manually via the LemonTree Addin.
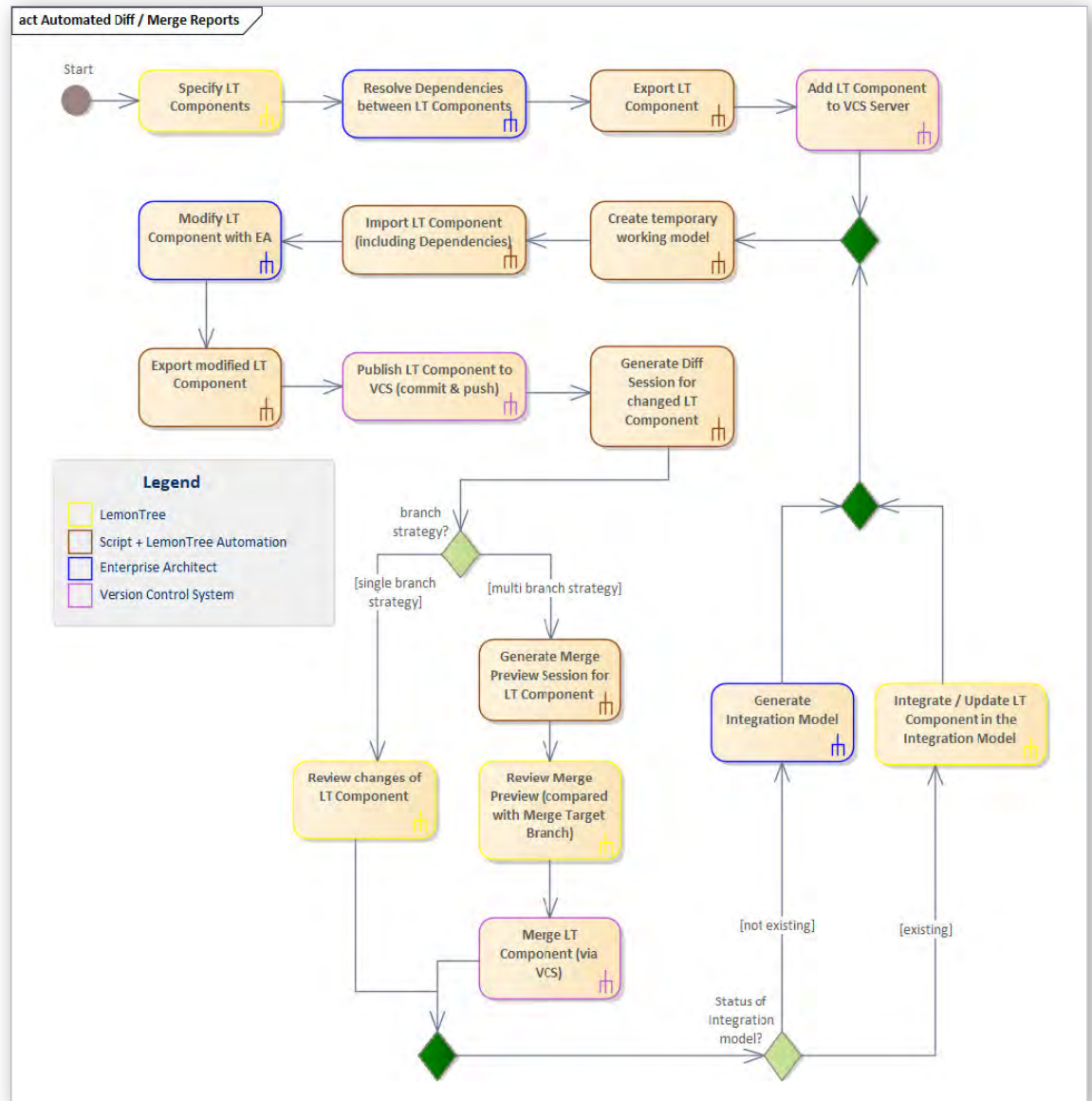

Figure 18: Automated creation of diff/merge reports

LEVEL 4: INTEGRATION INTO A (CENTRAL) OVERALL MODEL

In the highest expansion level (level 4), the integration model is updated automatically. For this, the central model (e.g. file server or database) is kept up to date via a script and the LemonTree Component Update by LemonTree Automation. This workflow extension corresponds to the comprehensive workflow from Figure 14.

CONCLUSIO

Working with LemonTree Automation and LemonTree Components enables fine-grained versioning of parts of an overall model. Familiar processes and tools from the area of Continuous Integration and DevOps are reused. As a tool, LemonTree Automation can be seamlessly combined with other common tools from development in order to take models into account in addition to code in familiar build processes.

https://www.lieberlieber.com/lemontree/de/automation-2/

The Authors

Philipp Kalenda
Senior Consultant

Dr. Konrad Wieland
CEO

Contact us at
welcome@lieberlieber.com