universität wien

2022W 290169-1 Prozessautomatisierung in der Geoinformationsverarbeitung

Prüfung linienhafte photogrammetrische Auswertung

Projektdokumentation

Dominik Knabe & Valerie Wöll

Februar 2023

Aufgabe des Workspaces

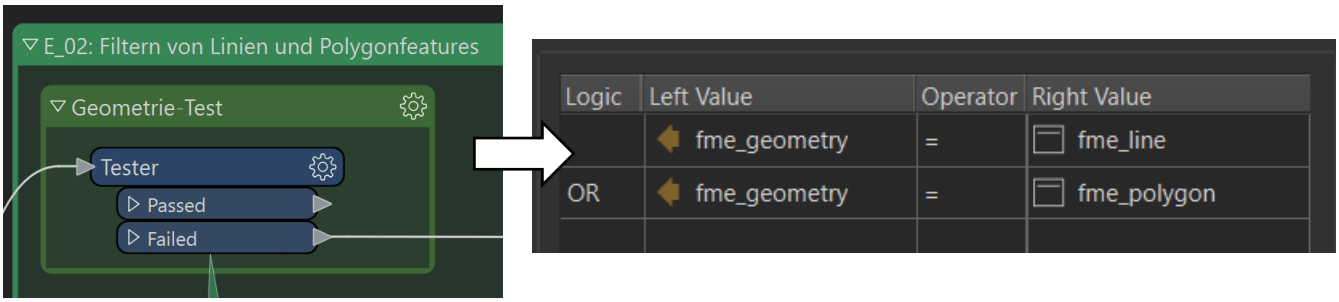
Ziel des Projektes ist es, einen Workspace in FME zu erstellen, der automatisiert Dachlinien von Gebäuden unter Berücksichtigung mehrerer Kriterien prüft. Dabei soll dieser auch für andere Dachlinien-Datensätze geeignet sein. Insgesamt wird auf 9 Regeln geprüft, die im Workspace parallel zueinander geprüft werden. Features, die die Prüfung nicht bestehen, sollen nach dem jeweiligen LevelName klassifiziert werden und im Format Bentley Microstation Design (V8) ausgegeben werden.

Einlesen und Zusammenführung der Daten

Die zu verarbeitenden Features sind als DGN-Dateiformat vorhanden, welche in CAD-Programmen generiert werden. Diese werden zu aller erst auf die dritte Nachkommastelle (Millimeter-Genauigkeit) gerundet um minimale Abweichungen und Fehler zu vermeiden. Der zweite einzulesende Datensatz ist eine Excel-Datei, welche die korrekte Bezeichnung der einzelnen Feature-Layer nach dem CityGRID Systemnamen beinhaltet. Nach erfolgreichem Einlesen wird die Namensgebung der Feature-Levels der Microstation mit den im Excel-Sheet angegebenen Namen für die Feature-Layers der CityGRID-Software abgeglichen. Dies erfolgt mit dem FeatureMerger“ über die Attribute „fme_feature_type“ und „Level_DGN“. Die korrekten Bezeichnungen sind nun im Attribut „Layer_CityGRID“ gespeichert. Daten, die auf Grund abweichender Bezeichnungen nicht zusammengeführt werden können, werden als Fehler **(E-31)** gefiltert und später gespeichert.

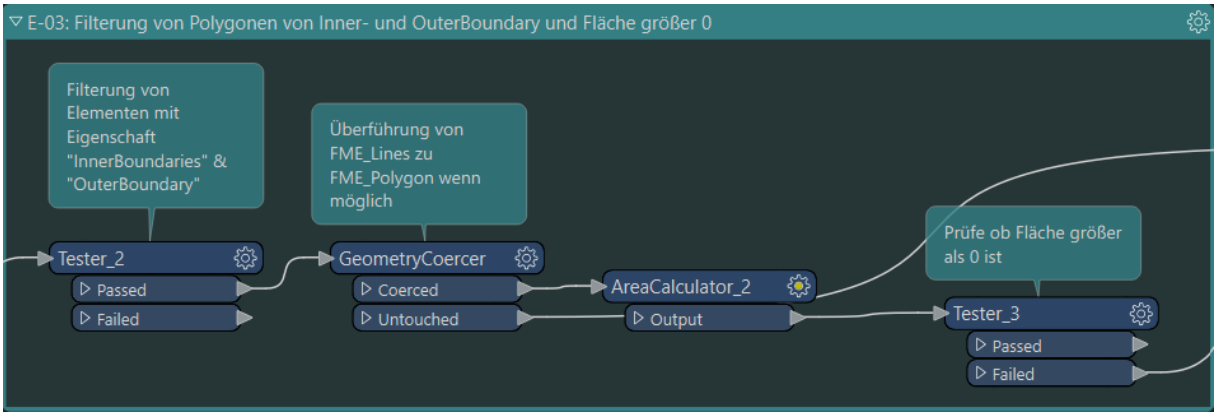
Es dürfen nur Linien und Polygonfeatures vorkommen (E-02)

Im Projekt sind nur Linien- und Polygonfeatures erlaubt. Daher wird der Datensatz nach diesen Charakteristiken mit einem „Tester“ gefiltert.



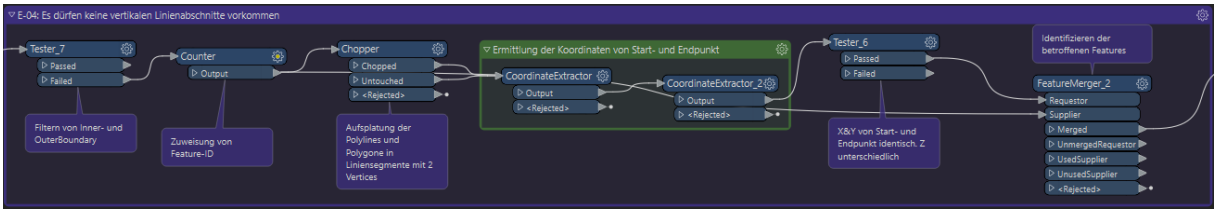
Für die Testung wurde das Attribut „fme_geometry“ verwendet. Alle Features, deren Attribut „fme_geometry“ den Wert „fme_line“ oder „fme_polygon“ enthalten, werden gefiltert. Im „Failed“-Output befinden sich alle Features, die den Test nicht bestanden haben und werden später abgespeichert.

Features der Layer „outerBoundary“ und „innerBoundary“ müssen geschlossene Polygone (Start- und Endpunkt sind identisch) mit Flächeninhalt größer 0 sein (E-03)



Um zu überprüfen, ob es sich bei den Features der Layer „innerBoundary“ und „outerBoundary“ um geschlossene Polygone mit Flächeninhalt größer null handelt, wird zunächst der Transformer „Tester“ verwendet, um die betroffenen Features zu filtern. Mit dem Transformer „GeometryCoercer“ werden alle Linien des Datensatzes, die mathematisch gesehen Polygone sind (Startpunkt = Endpunkt), als Polygone abgespeichert. Das ist wichtig, weil es Elemente geben kann, die Polygone sind aber im CAD-Programm als Linie definiert wurden. Diese sollten aber trotzdem in der Untersuchung berücksichtigt werden. In weiterer Folge werden mittels Transformer „AreaCalculator“ die Flächen der Polygone berechnet, um dann mittels Transformer „Tester“ zu überprüfen, ob diese Flächen größer null sind. Features die den Test nicht bestehen, befinden sich nun im Output-Port „Failed“ und werden später abgespeichert. Das Gleiche gilt für Features im „Untouched“-Output des „GeometryCoercers“, da diese keine Polygone sind.

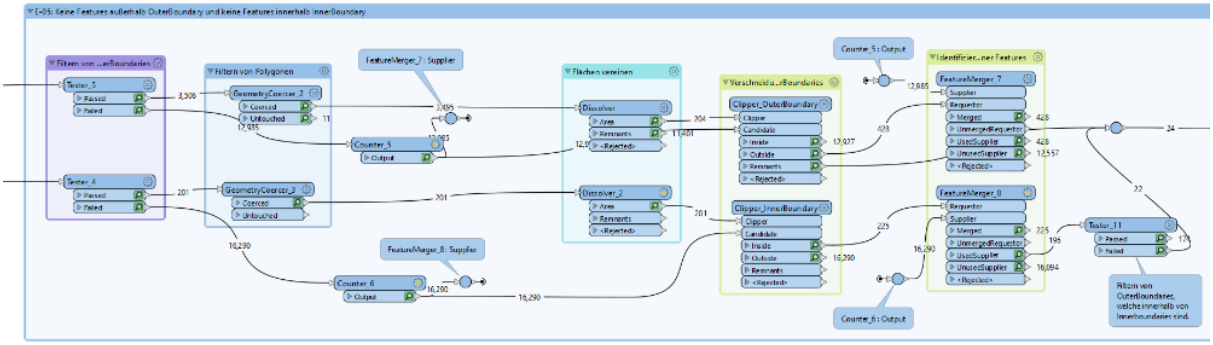
Es darf keine vertikalen Linienabschnitte, außer auf den Layern „outerBoundary“ und „innerBoundary“ geben (E-04)



Wie gewohnt werden zunächst die benötigten Layer gefiltert. Mit dem Transformer „Counter“ wird jedem Feature eine individuelle ID zugewiesen. Im Transformer „Chopper“ kann eingestellt werden in wie viele Segmente die Features aufgetrennt werden sollen. In unserem

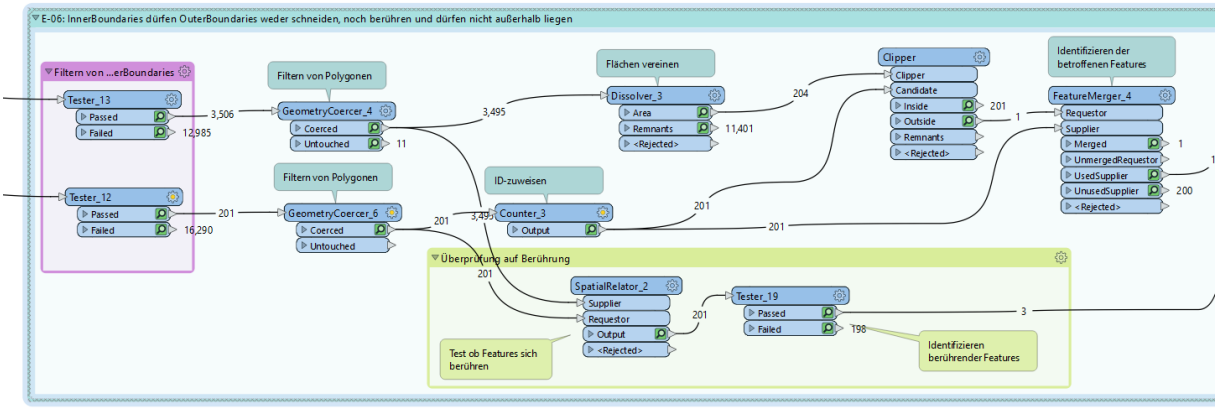
Fall wird hier für den Parameter „Maximum Vertices“ der Wert „2“ angegeben, um die einzelnen Liniensegmente zu erhalten. In weiterer Folge werden die Koordinaten (x,y,z) des Start- und Endpunkts (Index 0, -1) aller Liniensegmente der beiden Layer extrahiert. Somit kann nun mittels Transformer „Tester“ geprüft werden, ob diese Koordinaten denselben X- und Y-Wert aufweisen und zusätzlich, ob der Z-Wert unterschiedlich ist. Ist der Test positiv, ist das Segment vertikal. Über den zuvor erstellten Count, kann das Segment enthaltende Feature identifiziert werden und im Anschluss abgespeichert werden.

Es darf keine Features außerhalb der „outerBoundary“ und innerhalb der „innerBoundary“ Polygone geben. („outerBoundary“ Polygone innerhalb eines „InnerBoundary“ Polygons sind aber erlaubt und somit auch zugehörige Dachlinien) (E-05)



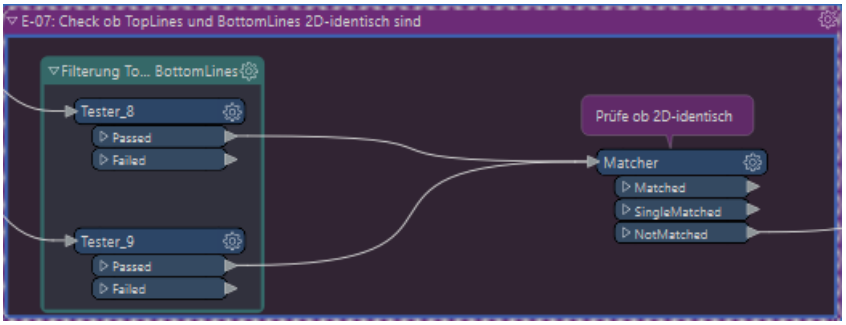
Um zu gewährleisten, dass sich keine Features außerhalb der „outerBoundary“ und innerhalb der „innerBoundary“ befinden, werden diese zunächst herausgefiltert. Der „Geometry Coercer“ stellt sicher, dass Linien, die mathematisch gesehen Polygone sind, auch als Polygone abgespeichert werden. Der „Dissolver“ vereinigt die Polygone. Die Clipper verschneiden nun alle Features außer den Canidates (jeweils „inner-„ oder „outerBoundaries“) mit den zuvor vereinten Boundaries. Alle Featuresegmente, die außerhalb der „outerBoundaries“ liegen befinden sich im „Outside“ Output. Jene, die innerhalb der „innerBoundaries“ liegen, im „Inside“ Output. Erneut werden die betroffenen Features über den „Count“ identifiziert und später abgespeichert. „outerBoundaries“, welche innerhalb der „innerBoundaries“ liegen, werden vor dem Speichervorgang herausgefiltert.

„innerBoundary“ Features dürfen „outerBoundary“ Features weder berühren noch schneiden oder außerhalb von „outerBoundaries“ vorkommen (E-06)



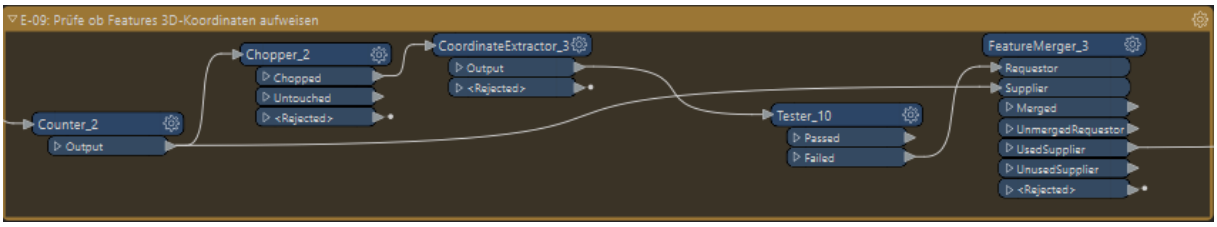
Diese Überprüfung erfolgt nach dem gleichen Prinzip wie der zuvor beschriebene Vorgang von Fehler E-05. Nur werden dieses mal nicht alle Features mit den „outerBoundaries“ verschnitten, sondern nur die „innerBoundaries“. Zusätzlich wird im unteren Bookmark geprüft, ob „innerBoundaries“ „outerBoundaries“ berühren. Dies erfolgt mit dem Transformer „SpatialRelator“. „innerBoundaries“, die „outerBoundaries“ berühren haben den Wert „1“ im Attribut „_related_suppliers“ und werden im darauffolgenden Tester gefiltert und später abgespeichert.

Features der Layer „breakTopLines“ und „breakBottomLines“ müssen 2D-identisch sein und dürfen sich nur in den Z-Werten unterscheiden (E-07)



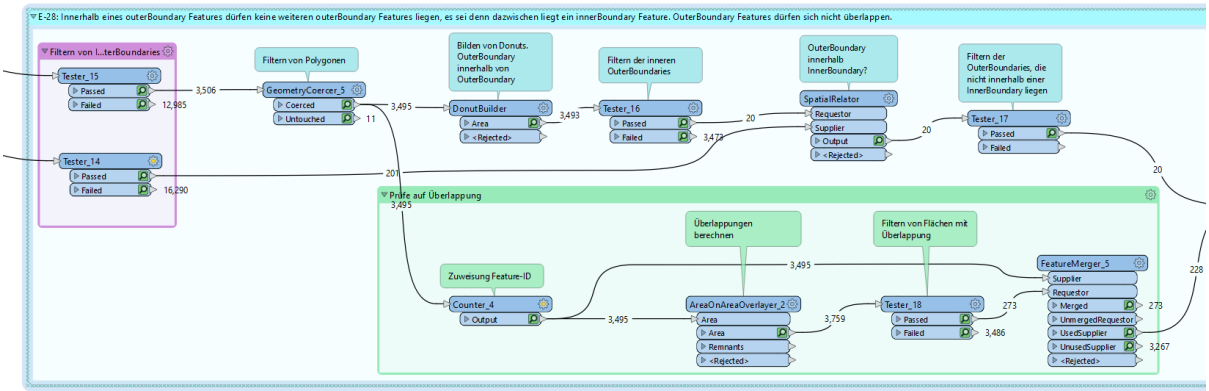
Zunächst wird nach den Layern „breakTopLines“ und „breakBottomLines“ gefiltert. Für die Abfrage, ob die Features 2D-identisch sind, eignet sich der Transformer „Matcher“. Hier kann direkt in den Einstellungen ein „Geometry Check“ durchgeführt werden. Dabei ist wichtig, dass die Einstellung „Match Geometry“ auf „2D“ gestellt ist. Die Features im Output-Port „NotMatched“ sind nicht 2D-identisch und kommen somit in die Fehlerauswertung rein.

Die Features müssen 3D Koordinaten (Z > 0) aufweisen, 2D Features sind nicht zulässig (E-09)



Um die Features auf ihr 3-Dimensionalität zu prüfen wird erneut die Kombination aus den Transformer „Chopper“ und „CoordinateExtractor“ verwendet. Diesmal wird beim „Chopper“ der Wert „1“ für die Einstellung „Maximum Vertices“ ausgewählt, um die einzelnen Vertexe zu erhalten. Nach Extraktion der Koordinaten wurde mittels „Tester“ geprüft, ob die Z-Koordinaten größer null sind (Theoretisch ist „0“ auch ein zulässiger Wert für die Höhe, jedoch bezieht sich der Datensatz auf Österreich, sodass es keine Höhen mit dem Wert „0“ geben kann). Im Nachgang werden die betroffenen Features anhand des „counts“ im „FeatureMerger“ identifiziert und in den Speichervorgang weitergeleitet.

Innerhalb eines „outerBoundary“ Features darf kein weiteres „outerBoundary“ Feature liegen, es sei denn ein „innerBoundary“ Feature liegt zwischen den beiden Features. „outerBoundaries“ dürfen sich auch nicht teilweise überlappen. (E-28)



Der „DonutBuilder“ erstellt zwei Flächen, wenn eine „outerBoundary“ innerhalb einer anderen “outerBoundary“ liegt. Die erste Fläche ist der Donut selbst und die zweite Fläche ist das Loch des Donuts. Über das Attribut „_hole_flag“ können Features identifiziert werden, die innerhalb eines anderen liegen (da diese die Umrandung des Loches des Donuts darstellen). Diese werden im Anschluss gefiltert und zusammen mit den „innerBoundaries“ in den „SpatialRelator“ geschickt, um zu überprüfen, ob jene „outerBoundaries“ widerum innerhalb einer „innerBoundary“ liegen. Alle “outerBoundaries“, für die das nicht gilt, werden in den Speichervorgang weitergeleitet.

Um zu überprüfen, welche „outerBoundary“-Polygone sich überlappen, werden diese in den „AreaOnAreaOverlayer“-Transformer geleitet. Jede Überlappung stellt im Output ein neues Feature dar und hat im Attribut „_overlaps“ den Wert „2“. Betroffene Features werden über den „count“ identifiziert und anschließend abgespeichert.

Es können nur Features auf vorgegebenen Layernamen verarbeitet werden (E-31)

➔ Siehe im Kapitel „Einlesen und Zusammenführung der Daten“

Strukturierung und Auswertung

Am Ende jeder Fehlerauswertung erhalten betroffene Features mit Hilfe eines „AttributeCreators“ das Attribut „Errorcode_“ mit dem jeweiligen Fehlercode (z.B. E28) als Attributwert.

Im letzten Arbeitsschritt wird zunächst die Datei "Fehlercodes.xlsx" eingelesen. Über das Attribut „Errorcode_“ der fehlerhaften Features und dem Attribut „Errorcode“ aus der eingelesenen Excel-Tabelle, werden im „FeatureMerger“ die Fehlerinformationen aus der Tabelle mit den Features vereinigt. Mit Hilfe des "FileNamePartExtractors" wird das Attribut "_rootname" erstellt, welches den Namen des eingelesenen .dgn Datensatzes enthält. Der "AttributeCreator" erstellt nun auf Basis der Fehlerinformationen aus der Excel-Tabelle den zugehörigen LevelName eines jeden Features, in das es abgespeichert wird. Über den darauffolgenden "Sorter" werden die Features nach ihrer Priority sortiert, sodass sie in der richtigen Reihenfolge abgespeichert werden. Der "AttributeKeeper" entfernt alle überflüssigen Attribute und stellt sicher, dass nicht veröffentlichte Systemattribute entfernt werden. Der "DGNStyler" färbt die Umrandungen der abgespeicherten Features rot. Im letzten Schritt werden alle betroffenen Features nach ihren zuvor definierten LayerNames als .dgn File abgespeichert. Das zuvor generierte Attribut „_rootname“ dient zur korrekten Bennenung des resultierenden Files, basierend auf dem eingelesenen Datensatz.