Change Log

Gennadi Heimann

29.12.2017

1 v0.0.1

@created on 14.11.2017
@finished on 21.12.2017

Der nächste Schritt wird erst freigegeben, wenn alle Abhängigkeiten der Komponenten geprüft sind. Beim Prüfen von der ausgewählte Komponente wird auch die SelectionCriterium geprüft, wieviel Komponenten in dem Schritt ausgewählt werden können.

Die ausgewählte Komponente wird in der CurrentConfig gesichert, damit das SelectionCriterium bewertet weden kann.

2 v0.0.2

@created on 4.12.2017
@finished on 19.03.2018
Die aktuelle Konfiguration behält gesamte Information über die hinzugefügte Komponente. Dazu gehören alle Abhängigkeiten und Einschränkungen und Komponente mit $< exclude >$ Markierung.

@created on 18.12.2017
@finished on 19.03.2018
Bei dem Abschluss der Konfiguration muss auf die Konsistenz den letzten Schritte geachtet werden. Es kann sein, dass in einem Schritt paar Komponente ohne weiteren Schritt und paar mit weiterem Schritt haben. (siehe szenario_3)

@created on 20.12.2017
@finished on 19.03.2018
Bei der Auswahl der gleichen Komponente innerhalb eines Schrittes muss diese Änderung erkannt werden und eine Info an den Webclient gesendet.

@created on 21.12.2017
@finished on 21.12.2017
Zu der $< ComponentOut >$ die $< componentId >$ und $< stepId >$ Parameter bei allen Status hinzufügen.

@created on 21.12.2017
@finished on 19.03.2018
Abwählen der Komponente muss auch implimentiert werden. Die Komponente wird auch aus der $< CurrentConfig >$ entfernt.

3 v0.0.5

@created on 13.12.2017
@finished on
@name

Jede Schritt in dem Konfigurator kann eine Abhängigkeit zu dem Schritt oder Komponente haben. Das bedeutet, dass geladene Schritt eine Komponente oder einen Schritt ausschließen oder fordern kann.

@created on 16.11.2018
@finished on
@name

Der Webclient soll prüfen, ob ein nächster Schritt exestiert. Wenn ja, kein Button für NextStep erstellen.

@created on 16.11.2018
@finished on
@name Auflösung der Abhängigkeit

Implementierung von $< exclude >$ und $< require >$ Abhängigkeiten zwischen der externen Komponenten. Es wird zwischen verschiedene Arten der Auflösung der Abhängigkeiten $< StrategyOfDependencyResolver >$ gesehen. Die Einstellung zu der Auflösung der Abhängigkeit wird direkt in dem $< Dependency >$ Edge gehalten.

- $< auto > \Rightarrow$ Die Abhängigkeit wird automatisch aufgelöst.
- $< selectableDecision > \Rightarrow$ Die Abhängigkeit durch die aktive Entscheidung der Anwender aufgelöst.

Interne Komponente sind die Komponente die einem Schritt gehören.
Externe Komponente sind die Komponente die verschiedenen Schritten gehören.

Bei der $< exclude >$ Abhängigkeit werde zwischen ausschließende und ausgeschlossenen Komponente unterschieden.
C1 - ausschließende Komponente
C2 - ausgeschlossene Komponent

Aussage 1: Komponente C1 schließt die Komponente C2 aus der Konfiguration aus. Das bedeutet, dass der Komponente C2 darf nicht ausgewählt werden, wenn die Komponente C1 davor ausgewählt war. Das war eine $< forward >$ Auflösung der Abhängigkeit.

Aussage 2: Komponente C1 schließt die Komponente C2 aus der Konfiguration aus. Aber im Gegensatz zu der Aussage 1 wird die Komponente C2 vor

C1 ausgewählt. Das bedeutet, dass die Komponnte C2 aus der Konfiguration entfernt wird und rekursiv nach anderen Abhängigkeiten gesucht und aufgelöst. Diese Auflösung wird als $< reverse >$ Auflösung der Abhängigkeiten genannt.

@created on 23.11.2018
@finished on
@name

Erkennung der Konfigurationsverlauf.
Es gibt zwei Konfigurationverläufe:

- $< configurationCourse = sequence >$
- $< configurationCourse = substitute >$

Der Konfigurationsverlauf wird in dem Vertex Config abgespeichert.

@created on 23.11.2018
@finished on
@name

Die Abhängigkeiten zu der aktuelle Konfiguration hinzufügen.

@created on 23.11.2018
@finished on
Die Logik Ausdenken mit der die Info zu der ausgewählte Komponente veranschaulicht.

3.1 v0.0.6

@created on 23.11.2018
@finished on

Visualisierung der Abhängigkeit (behavior)