Automatisiertes Engineering

Mit Wizards schneller zum Erfolg

www.copadata.com
sales@copadata.com

zenon
do it your way

Content

Der „zenon automotive generator“ (zag) ist ein Wizard zur automatisierten Analyse von SPS-Daten und Umsetzung von Visualisierungsprojekten. Zum Einsatz kommt er in erster Linie in der Automobilindustrie, die traditionell stark auf standardisierte Komponenten und Wiederverwendung setzt.

Die Automatisierung von Engineering-Prozessen ermöglicht eine deutliche Zeit- und Kostenersparnis in der Projektumsetzung. Die oft knappen Zeitvorgaben lassen sich damit einfacher einhalten. Simple und wiederkehrende Aufgaben können vom Projektanten an den Wizard ausgelagert werden – so bleibt mehr Zeit für anspruchsvolle Tätigkeiten und das Risiko von Fehlprojektierungen ist minimal.

In diesem White Paper werde ich ein paar Aspekte zur Erstellung eines Wizards wie dem „zenon automotive generator“ beschreiben. Neben der technischen Umsetzung sind auch ein paar Vorüberlegungen aufgeführt und Hintergründe beschrieben.

1. Das Programmierinterface als Basis für den Generator

Das Programmierinterface (API) von zenon wird gerne für aufgabenspezifische Erweiterungen genutzt. Neben dem API der zenon Runtime gibt es auch eine entsprechende Schnittstelle zum zenon Editor, die für die Automatisierung von Engineering Aufgaben zur Verfügung steht. Beispiele hierzu sind die mit dem zenon Editor mitgelieferten Wizards. Die Offenheit von zenon erlaubt es jedem Anwender des Editors eigene Wizards zu erstellen oder bestehende projektspezifisch anzupassen. Dafür stehen die integrierten Programmiersprachen wie VBA oder VSTA (C# oder VB.NET) zur Verfügung. Außerdem existiert eine Vorlage eines Wizards für die Programmierung mit dem Microsoft Visual Studio.

Bei der Auswahl der Programmiersprache kann auch „Know-how Schutz“ eine Rolle spielen. So kann der Wizard in einer Art „Open Source“ zur Anpassung von Dritten erstellt werden, oder er wird in kompilierter Form verwendet.

2. Die Vorteile eines Wizards

Ein großer Vorteil beim Einsatz von Wizards ist die verkürzte Projektierungszeit. Die Laufzeit der automatischen Generierung beträgt meist nur einen Bruchteil im Vergleich zur benötigten Zeit bei manueller Projekterstellung. Hinzu kommt noch eine weitere Zeitersparnis: der Generator kann die Informationen deutlich schneller lesen und nach den hinterlegten Regeln interpretieren als dies ein Mensch machen könnte. Um den „Mehrwert“ des Wizards zu

bestimmen sollte dieser zeitlichen Vorteil – und damit die dadurch eingesparte Arbeitszeit – ins Verhältnis zum zeitlichen Aufwand für die Erstellung des Wizards gesetzt werden. Mit der Anzahl der generierten Projekte besteht ein linearer Zusammenhang:

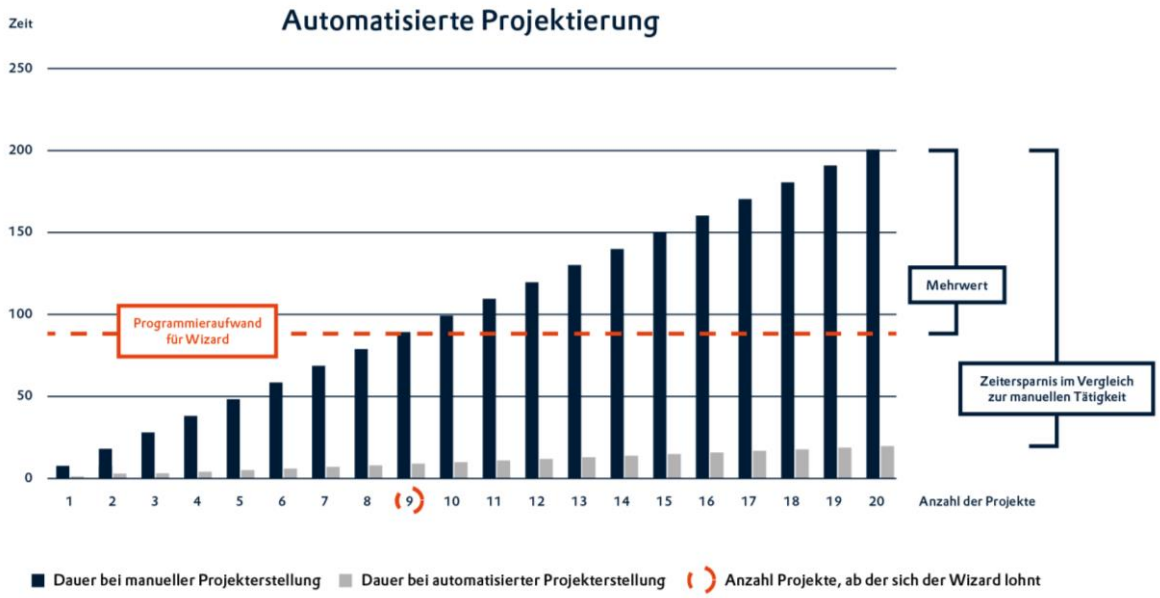

Abbildung 1: Zeitersparnis durch Automatisierte Projektierung

In diesem Diagramm wird von einem Verhältnis von 1 zu 10 ausgegangen: während die manuelle Tätigkeit 10 Zeiteinheiten benötigt, braucht der Wizard nur 1 Zeiteinheit. Bei 10 Projekten ergibt sich dadurch eine Zeitersparnis von 90 Zeiteinheiten. Verbindet man nun die Aufwandsabschätzung zur Codeerstellung mit dieser Kurve erhält man die Information ab wie vielen Projekten sich ein solcher Wizard lohnt. In diesem Beispiel ergibt sich ein Mehrwert durch die Erstellung eines Wizards ab dem zehnten Projekt.

Die mit einem Wizard eingesparte Arbeitszeit wird häufig als der größte Vorteil gesehen, der auch direkt messbar ist. Es gibt jedoch noch weitere Vorteile, die zwar nicht unmittelbar messbar sind, aber eine positive Auswirkung auf die Qualität der Projekte haben:

- Der Wizard wird die Projekte immer exakt nach den hinterlegten Regeln erzeugen. Fehler, die sich durch manuelle Arbeiten ergeben, können vollständig vernachlässigt werden.

- Die Anwender des Wizards benötigen kein tiefgreifendes Know-how über das Projekt oder die eingesetzten Systeme. Abhängig von der Bedienoberfläche reicht häufig eine kurze Anleitung über die Ausführung.
- Der Wizard liest für die Auswertung der Informationen eine Datenquelle aus. Kann diese nicht richtig interpretiert werden, liegt dies an fehlerhaften Informationen. Somit führt Wizard eine zusätzliche Qualitätsprüfung der Datenquelle durch.

3. Regeln der Projektgenerierung: Wer? Wie? Was? Wo?

Vor dem eigentlichen Programmieren müssen die „Regeln“ der Projektgenerierung so exakt wie möglich definiert werden. Diese beschreiben welche Informationen wo gefunden werden, wie diese zu interpretieren und anzuwenden sind. Solche Regeln kommen auch bei manuell erstellten Projekten zur Anwendung. Häufig sind diese jedoch nicht so klar dokumentiert, dass sie für die Erstellung des Generators übernommen werden können. Beim Erstellen eines Wizards werden diese Regeln programmiert, sodass sie bei der Ausführung exakt angewendet werden. Der Wizard dient hierbei als „Interpreter“ einer Datenquelle und als Übersetzer in ein zenon Projekt.

Nach dem Festlegen dieser Regeln können sie für die Übersetzung in den eigentlichen Programmcode in einem Flussdiagramm dokumentiert werden. Diese Grafik beschreibt die einzelnen Schritte zur Abarbeitung durch den Wizard.

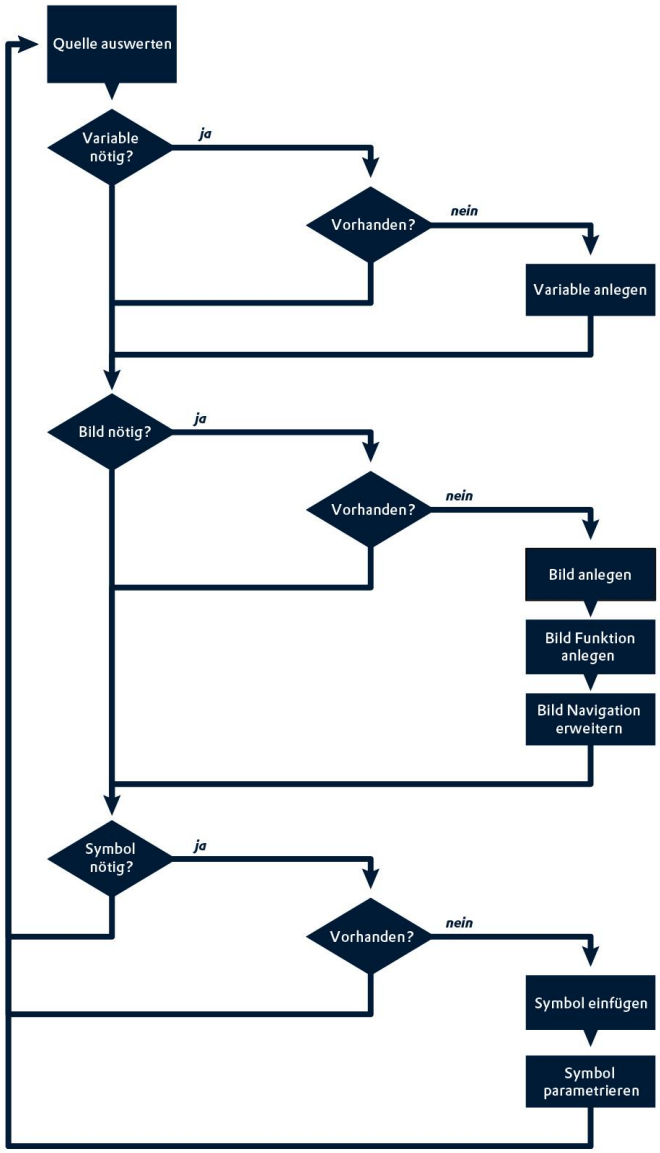

Abbildung 2: Beispiel für ein Flussdiagramm als Grundlage eines Wizards

Für das eigentliche Auslesen der Datenquelle ist ein entsprechender Aufwand zu berücksichtigen. Für den Zugriff auf Excel-Dateien findet man viele Code-Beispiele im Internet. Geht es aber darum, auf Daten in einer SPS-Programmiersoftware zuzugreifen, ist manchmal eine Recherche über die Zugriffsmethoden und die Datenspeicherung notwendig.

Neben der Beschreibung der Aktionen des Wizards sollte der Programmierer auch eine funktionale Bedienoberfläche vorsehen. Während der Wizard im Hintergrund seine Arbeit verrichtet wird damit der Bediener über die Aktionen informiert. Der Anwender kann die Oberfläche nutzen um eventuell notwendige Parameter zu konfigurieren, oder sie informiert ihn über den Fortschritt der Generierung.

4. Einfache Erstellung des Generators

Für die Erstellung des Codes stehen dem Programmierer die unterschiedlichen zenon Objekte zur Verfügung. Diese sind in der Online Hilfe dokumentiert. Die jeweiligen Eigenschaften („Properties“) sind auch in der eingebetteten Hilfe von zenon zu finden. Über diese Eigenschaften können die jeweiligen zenon Objekte angepasst werden.

Für die Erzeugung der unterschiedlichen Projektteile werden die Objekte entsprechend angelegt und mit Werten belegt. Variablen werden beispielsweise mit diesem VBA Code erzeugt:

```
Set zVar = MyWorkspace.ActiveDocument.Variables.CreateVar(a, b, c, d)
```

Hierbei entsprechen die Platzhalter folgenden Objekten:

a = der Name der zu erzeugenden Variable

b = das Treiber Objekt auf dem die Variable basieren soll

c = der Treiberobjekttyp der Variable

d = der Datentyp der Variable

Diese Informationen entsprechen den gleichen Parametern die auch beim Anlegen einer Variable im zenon Editor einzugeben sind:

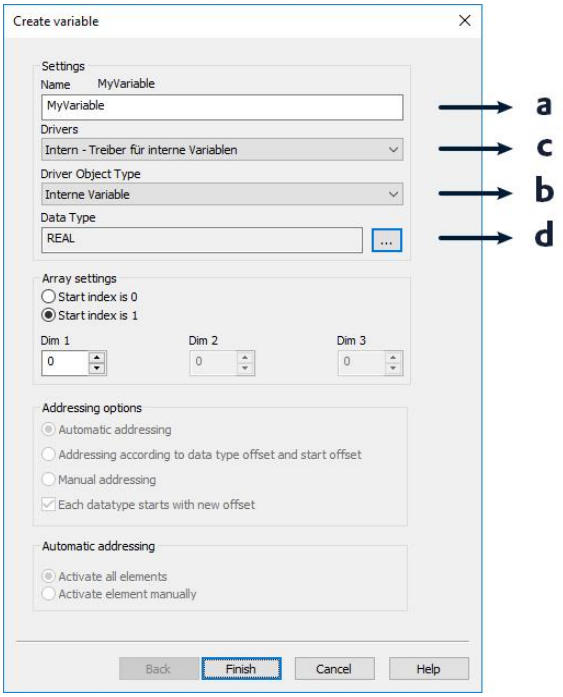

Abbildung 3: Parameter einer Variable in zenon

Der Parameter a im obigen Code entspricht dem Feld „Name“ in diesem Dialog. Der Parameter b dem Feld „Driver Object Type“, der Parameter c der Combobox „Driver“ und der Parameter d dem „Data Type“.

Weitere Eigenschaften dieser Variable werden über die „DynProperties“ der jeweiligen Objekte angepasst. Um beispielsweise die Variablenkennung zu ändern, ist die jeweilige Eigenschaft zu ermitteln und diese zu konfigurieren. Der Name dieser Eigenschaft kann im zenon Editor aus der eingebetteten Hilfe ermittelt werden. Klickt man auf die Variablenkennung einer Variable wird hier unterhalb des eigentlichen Hilfetextes der Name der Eigenschaft (hier „Tagname“) angezeigt. Dieser kann mit

```
zVar.DynProperties(„Tagname“) = „automotive“
```

entsprechend verändert werden.

Für einzelne, konstante Parameter stehen dem Programmierer bei der Codeerstellung diese als Konstanten zur Verfügung. Ein Beispiel hierzu sind die unterschiedlichen Funktionstypen bei der Erstellung einer neuen zenon Funktion:

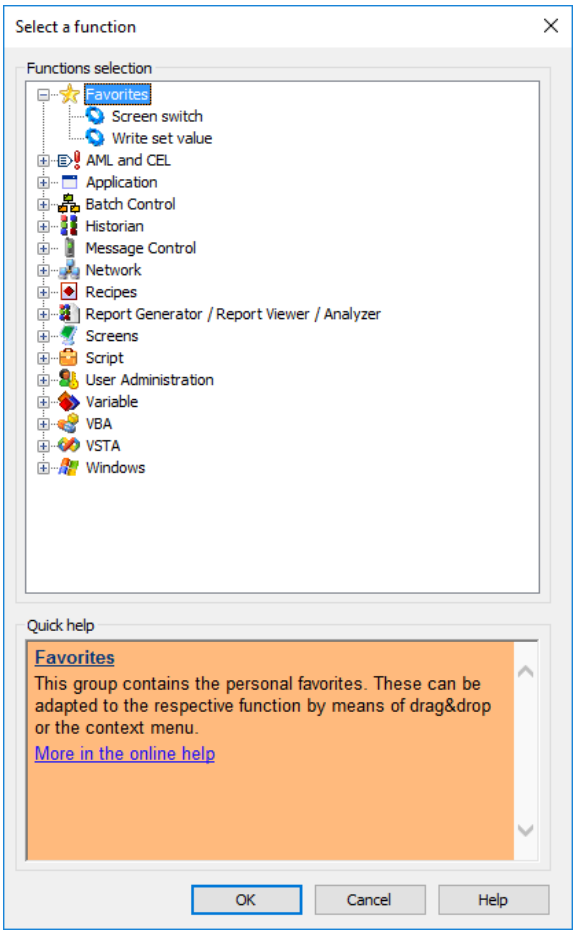

Abbildung 4: Auswahlfenster der zenon Funktionen

Erzeugt man eine neue Funktion mit Code wird neben dem Namen der Funktion der Funktionstyp als konstanter Parameter übergeben.

```
set myFunc = myworkspace.ActiveDocument.RtFunctions.Create("name",
```

Create(*bsName As String*, ***tpFuncType As*** **tpFunctionTypes**) As RtFunction

- tpActivateAlarm
- tpActivateInput
- tpAlarmAck
- tpAlarmAckBlink
- tpAlarmgroupOnOff
- tpAlarmsGroupsClasses

Abbildung 5: Erzeugen einer Funktion mit Code

Anschließend können abhängig vom Funktionstyp die zugehörigen Eigenschaften wieder über die „DynProperties“ konfiguriert werden.

```
Set myFunc = myWorkspace.ActiveDokument.RtFunctions.Create(“func-name”,
                                tpPicture)

myFunc.DynProperties(“Picture”) = “screename”
```

Mit diesem Code wird eine neue Funktion mit Namen “func-name” vom Typ Bildumschaltung (= Konstante tpPicture) erzeugt. Als Parameter wird das Bild mit Namen „screenname“ verwendet.

Mit diesem Vorgehen kann der Code des Wizards schnell erstellt werden. Die Erzeugung und Parametrierung der zenon Objekte folgt immer diesem Muster.

Um die Laufzeit des Generators stabiler zu gestalten sollten einige Sicherheitsabfragen integriert werden. So können beispielsweise keine neuen Variablen mit dem Namen von bereits bestehenden Variablen angelegt werden. Deshalb empfehle ich, entsprechende Abfragen zu implementieren und darauf zu reagieren.

5. Schneller Einstieg – schnelle Erfolge

Mit den unterschiedlichen Code Beispielen in der zenon Online Hilfe und im COPA-DATA Forum sind die ersten Schritte zur Erstellung eines Generators schnell erledigt. Mit der Definition der Regeln und der Umsetzung im Wizard erhält der Programmierer in kurzer Zeit eine erste Version. Damit kann der Anwender erste, positive Erfahrungen sammeln.

Mit dem Einsatz von Wizards kann in der Projekterstellungsphase sehr viel Zeit eingespart werden. Fehler, die bei immer gleichbleibenden Aufgaben entstehen, werden durch die Automatisierung der Vorgänge verhindert.