

Nutzung von eigenen Datentypen in UML (Unified Modeling Language)

Viele *Rhapsody®* Anwender stoßen nach kurzer oder längerer Zeit auf die Frage wie sie eigene Datentypen in einem UML Modell verwenden können. Im Folgenden möchten wir einen Weg aufzeigen wie das möglich ist.

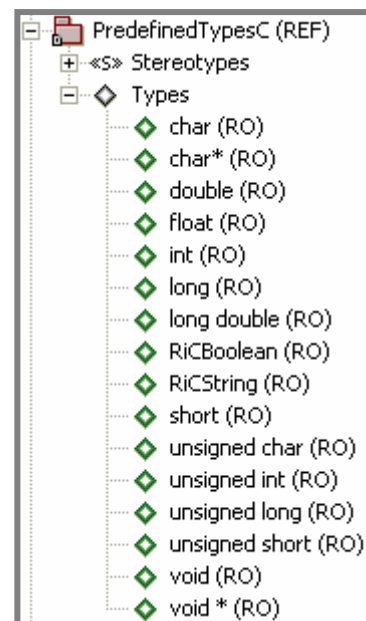
In einem neu angelegten Modell stellt *Rhapsody®* zuerst einmal Standard Datentypen zur Verfügung die der jeweiligen Sprache entsprechen für die Code generiert werden soll.

Demzufolge bietet *Rhapsody® in C* folgende Datentypen als default an:

char, short, int, long, double, long double, float, unsigned char, unsigned short, unsigned int, unsigned long, void, void *, char * und die *Rhapsody®* Typen **RiCBoolean** und **RiCString**.

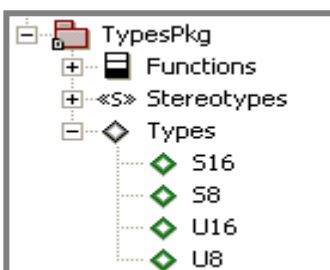
Diese Datentypen reichen für viele Zwecke aus, haben aber auch Nachteile. Wenn man zum Beispiel sein Modell und den daraus generierten Code weitgehend unabhängig von Compiler und CPU machen möchte. In diesem Fall sind z.B. Datentypen sinnvoll, die auf allen Plattformen konsistente Größe und Verhalten haben.

Ein anderer Grund kann MISRA konformer Code sein. In verschiedenen Fällen kommt man also um eigene Datentypen nicht herum.



In der Notation ANSI-C wird das durch das Anlegen von typedefs erreicht. Zum Beispiel „U8“, „S8“, „U16“, „S16“ usw. wobei U für unsigned, S für signed und die Zahl für die Anzahl der Bits steht.

Auch in *Rhapsody®* ist es sehr komfortabel möglich mit eigenen Datentypen zu arbeiten. Hierfür bietet es sich an die neuen Datentypen alle zusammen in einem gemeinsamen Package zu definieren. Für jede Plattform würde man ein solches Package definieren mit den für diese Plattform entsprechend abgeleiteten Datentypen.



Types werden als UML Type wie folgt definiert: In einem Package “Add New” und dann “Type” anwählen. Dann unter “General” als “Kind” “Typedef” auswählen und im Tab “Details” den richtigen Basistype auswählen. Also für U8 ein “unsigned char” und für S16 ein “short”.

Wenn das gemacht wurde können diese Datentypen für Attributen, Variablen, Argumente und Returnvalues von Operationen ausgewählt werden. Die ursprünglichen Standardtypen sind weiterhin vorhanden. *Rhapsody®* bietet 2 Properties mit denen auch das geändert werden kann:

General:Model:CommonTypes “TypesPkg”
General:Model:DefaultType “TypesPkg::U16”

Nutzung von eigenen Datentypen in UML (Unified Modeling Language)

Alle in dem ersten Property angegebenen Packages bzw. den darin enthaltenen Type Definitionen werden in den *Rhapsody*® Type Dialogen aufgelistet. Wenn die Property leer ist werden die PredefinedTypes(C) aufgelistet.

Der erste Eintrag eines eigenen Datentype bewirkt, dass die Standard Datentypen NICHT mehr gezeigt werden. Sollen diese weiterhin angezeigt werden, dann müssen die "PredefinedTypes(C) Packages auch mit aufgenommen werden. (\$ALL bedeutet, dass alle Typen im Modell gelistet werden). Die angegebenen Packages müssen mit dem vollständigen Pfad in Form einer Liste durch ";" getrennt aufgenommen werden.

Die zweite Property enthält den Datentype den *Rhapsody*® als Default anbietet. Achtung! Vollständiger Pfad, also Package::Type angeben.

Jetzt gibt es aber noch die Animation zu beachten. Die *Rhapsody*® Animation kann die neuen Datentypen nicht darstellen. Um auch die korrekte Darstellung zu ermöglichen müssen Funktionen zur Verfügung gestellt werden, die eine Konvertierung von Datentype zu String und zurück vornehmen. Die Namen dieser Funktionen sind abzuleiten vom Datentype. Z.B: `serialize<Datentype>` und `unserialize<Datentype>`. Für ein U8 müssen also folgende Funktionen implementiert werden:

```
char * serializeU8( U8 n);
```

Diese Funktion reserviert Speicher mit `malloc()` für ein String und schreibt den Inhalt der Variablen dort hinein. *Rhapsody*® wird den Speicher wieder freigeben.

```
U8 unserializeU8( char * s, U8 n);
```

Bekommt einen String, extrahiert den Inhalt und schreibt ihn zurück in die Speicheradresse der Variablen.

Diese Funktionen werden natürlich nur benötigt wenn Animation genutzt wird und Animation eingeschaltet ist. Aus diesem Grund benötigen wir noch einen Stereotype `<AnimationOnly>`, das folgende Properties setzt:

```
C_CG:Operation:ImplementationEpilog  
"ifdef _OMINSTRUMENT"  
  
C_CG:Operation:ImplementationProlog  
"endif // _OMINSTRUMENT"
```

`_OMINSTRUMENT` wird durch *Rhapsody*® gesetzt, wenn Animation aktiviert ist. Dieser Mechanismus bewirkt nun, dass die Funktionen nur dann mit kompiliert werden, wenn Animation aktiv ist.

Ein Modell mit einem kleinen Beispiel finden Sie auf unserer Homepage unter folgendem Link:

www.willert.de/models-source-code/flat/116

Diese und mehr Informationen finden Sie in unserem UML-Forum: www.willert.de/uml-forum

Auf diese Weise können mit *Rhapsody*® auf Unternehmensebene, Projektebene oder Mitarbeiterebene individuelle oder einheitliche Datentypen genutzt werden.

Viel Spaß mit *Rhapsody*®

Willert Software Tools GmbH

www.willert.de



UML logo are trademarks or registered trademarks of the Object Management Group, Inc. in the United States and other countries.

Rhapsody® is a registered trademark of Telelogic (an IBM Company)