

# 10 Praxistipps für den UML-Einstieg

**WILLERT.**  
pioneers in embedded software engineering



## **Trainings besuchen 01**

Mit intensiven Trainings oder Workshops erhalten Sie einen ersten Praxiseinstieg in UML. Diese müssen allerdings auf Embedded Software zugeschnitten sein. 90% aller UML-Schulungen sind für Embedded Entwickler nicht geeignet und damit Fehlinvestitionen.

## **Nicht halbherzig einsteigen 02**

Risiko ist ihr Feind. Deshalb sollten Sie den Einstieg in kleinen Schritten - bezogen auf Ihr Projekt - durchführen. Aber er sollte für ein bis drei Arbeitsplätze konsequent durchgezogen werden: mit geeigneter Schulung, vernünftigen Tools und durchgängig bis zur Implementierung auf der endgültigen Zielhardware. Auf dieser Erfahrungsbasis können Sie UML in Ihren Projekten immer mehr ausweiten und damit das Risiko klein halten.

## **Über Feedback lernen 03**

Rückmeldungen zu Ihrer Arbeit erhalten Sie nicht nur durch externe Projektcoaches (siehe Pkt 9), sondern auch durch den konsequenten Einsatz von UML bis hinunter zur Implementierung (fragmentarische Target-Implementierung). CASE-Tools mit Codegenerierung geben Ihnen über den erzeugten Code direkt Feedback, wie gut Sie UML schon beherrschen. Lassen Sie diesen Code auch unbedingt auf der Zielhardware laufen, um Rückmeldung über das Laufzeitverhalten zu bekommen.

## **Versprechungen der Toolhersteller hinterfragen 04**

Hinterfragen Sie die Versprechungen der Toolhersteller kritisch, bspw. der schnellen Anpassung der Codegenerierung an Ihr System (CPU, Compiler, RTOS). Lassen Sie sich eine lauffähige Lösung zeigen und verlangen Sie verlässliche Aussagen über das Footprint des generierten Codes. Wenn es keine konkreten Aussagen zu Ihrer Zielplattform gibt, kalkulieren Sie mehrere Monate Aufwand für die Evaluierung ein.

Willert Software Tools GmbH  
Hannoversche Str. 21  
DE - 31675 Bückeburg  
Email: [info@willert.de](mailto:info@willert.de)  
Tel.: +49 5722 9678 - 60

## **Grenzen der UML kennen 05**

UML ist keine eierlegende Wollmilchsau. So kann sie keine zeitkontinuierlichen Applikationsteile abbilden. Verwenden Sie hier Reglermodelle, abgebildet mit MathWorks MATLAB, ETAS ASCET-SD oder National Instruments MATRIXx. Rechnen Sie damit, dass Sie auch mit dem generierten C- oder C++ - Code noch Arbeit haben. Zudem sind nicht alle UML-Konstrukte einsetzbar, wenn Systeme mit Ressourcen haushalten müssen. Welches Konstrukt sich wie auf Laufzeit oder Codegröße auswirkt, zeigt am besten der generierte Code.

## **Brüche zwischen Analyse, Design und Implementierung vermeiden 06**

Diese Brüche sind die häufigsten ROI-Killer. Erst das Erstellen von Produktionscode in Verbindung mit Reverse-Engineering (Einbindung von vorhandenen C-Sourcen) und Roundtrip-Engineering (Änderungen auf C-Ebene, die automatisch in das Modell übernommen werden) machen den UML-Einsatz erfolgreich. Sind UML-Design und Code identisch, ist das System verständlich und Komponenten sind erneut einsetzbar.

## **Alten Code einbinden 07**

Natürlich kann alter Code mit einem UML-Modell wieder verwendet werden. Dies geschieht auf Ebene der C-Schnittstellen. Versuchen Sie nicht den Code 1:1 in das Modell zu übertragen oder nachträglich objektorientiertes Denken in prozeduralen Code zu bringen. Konzentrieren Sie sich nur auf die Schnittstellen zum alten Code, die Sie in das Modell übertragen sollten.

## **Ein Laufzeitsystem einsetzen 08**

Das Architekturdesign drückt das Taskverhalten des Systems aus. Sorgen Sie dafür, dass die modellierte Architektur mit dem Verhalten auf der Hardware identisch ist. Dieses erhält man nur durch den Einsatz eines Embedded Betriebssystems oder eines Laufzeitsystems mit ähnlichen Eigenschaften.

## **Mit Coaching die Einführungszeit verkürzen 09**

Projektbegleitendes Coaching sorgt für den reibungslosen Übergang vom Training in die Praxis. Dies in wenigen Tagen oder Wochen. Coaches begleiten Ihre ersten Schritte und sorgen dafür, dass die grundsätzliche Richtung stimmt.

## **Ans Testen denken 10**

Ein gutes CASE-Tool unterstützt Sie in allen Projektphasen. Sie können Requirements abbilden, Analyse und Design durchführen, sowie natürlich Code generieren. Vervollständigen Sie Ihr UML-Modell um die Testfälle und erzeugen Sie den Code dafür aus dem Modell. Damit erreichen Sie bereits im Ansatz eine höhere Testabdeckung als bisher.