

TIMING-MODELLE UND -ANALYSEN
RICHTIG EINSETZEN - TEIL I

Echtzeit-Methodik für AUTOSAR-Serienentwicklungen

Die in diesem Beitrag vorgestellte Methodik, der AUTOSAR-Standard und das Werkzeug SymTA/S unterstützen heterogene Timing-Modelle durchgängig über den gesamten Entwicklungszyklus. Die Methodik ist durchgängig und sowohl für neue als auch für bereits existierende Systeme einsetzbar. Das Ergebnis ist eine frühzeitige Optimierung und umfassende Bewertung der Realisierbarkeit auf Basis der aktuellen Informationen, die sich klar darstellen, kommunizieren und argumentieren lässt, insbesondere gegenüber dem Projektmanagement, Software-Zulieferern und OEM-Kunden.

AUTOSAR ist heute endgültig als Standard für Steuergeräte-Entwicklungen etabliert und wird in Serienprojekten genutzt. Aus Sicht der Software-Entwicklung hat AUTOSAR bereits eine enorme Produktivitätssteigerung erreicht. Für das gesamte ECU-Projekt gibt es jedoch eine Reihe weiterer Anforderungen wie z. B.:

- Realisierbarkeit frühzeitig bewerten durch virtuelle Integration der noch zu implementierenden Softwarekomponenten. Dabei spielt das Thema „Echtzeitfähigkeit“ eine zentrale Rolle.
- Sicherheit/Verfügbarkeit garantieren, Plattform-Erweiterbarkeit gewährleisten (über Integrationsstufen und Produktzyklen), dadurch Vermeiden später und kostspieliger Design-Änderungen.
- Standardisierte Dokumentation der Echtzeiteigenschaften (Stichwort Lastenheft).

Diese Anforderungen lassen sich nur über eine Systemintegration erfüllen, die ECU-Ressourcen und die verfügbare Rechenzeit optimal nutzt. Dazu ist es notwendig, die Funktionsarchitektur in eine geeignete SW-Architektur umzusetzen, die RTE- und OS-Konfiguration, insbesondere den Schedule optimal zu bestimmen, und die Basis-Software ressourcenschonend zu integrieren. Timing-Analysen unterstützen dabei, indem sie eine nachvollziehbare und argumentierbare Bewertung der jeweils anstehenden Entwurfsentscheidungen liefern. Geeignete Timing-Modelle

ermöglichen dabei die standardisierte Dokumentation bis hin zur eindeutigen Timing-Anforderung an Zulieferer.

Im Folgenden werden elementare Timing-Analyse-Schritte beschrieben, die sich an der industriellen Entwicklungs-Praxis orientieren, mit dem AUTOSAR-Standard beschreibbar sind und sich der verfügbaren Timing-Analyse-Tools bedienen. Die wichtigsten Analyseaspekte sind:

- CPU-Last pro Funktions- bzw. Software-Komponente, sowie verfeinert die Lastanteile und Zykluszeiten der in den Software-Komponenten enthaltenen Runnables,
- Zeitbedingungen für Wirkketten (Event Chains) aus der Funktionsentwicklung,
- Zusammenhänge zwischen funktionalen Wirkketten und der Softwarearchitektur und
- Systemkonfiguration als Summe aller integrierten Funktionen und deren Scheduling.

CPU-Auslastung

Die einfachste aller Analysen ist die Festlegung der CPU-Last pro Applikationsfunktion. Aus der Summe ergibt sich die Auslastung der CPU. Die natürliche Grenze ist 100%, in der Praxis werden ECU-Projekte oft mit Grenzwerten von 50% bis 70% begonnen, denn man benötigt Reserven für die Basissoftware, spätere Integrationsstufen und Produkt-Updates. Bei Multi-Core-Systemen lässt sich nach derselben Methode die Last pro Core bestimmen und dabei idealerweise bereits eine gleichmäßige Auslastung erreichen.

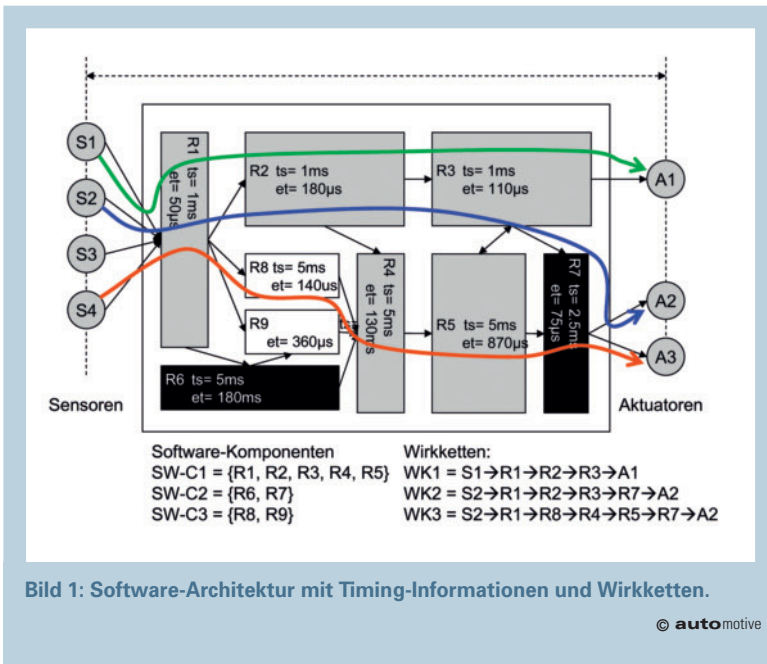


Bild 1: Software-Architektur mit Timing-Informationen und Wirkketten.

© automotive

für die elementaren Timing-Parameter der einzelnen Software-Teile fest, was den Aspekt der Echtzeit von nun an explizit im Projekt planbar und überprüfbar macht.

Virtueller Task-Schedule

Nun wird aus den Runnables ein erster „virtueller“ Task-Schedule gebildet. Dabei folgen man einem für das Projekt geeigneten Schedulingkonzept, z. B.:

- Zusammenfassen von Teil-Funktionen derselben Zykluszeit in jeweils eine Task.
- Je kürzer die Zykluszeit, desto höher die Task-Priorität. Beide Informationen lassen sich in RTE- und OS-Config. standardisiert nachhalten.

Nun überprüft man anhand dieses virtuellen Task-Schedules eine essenzielle Echtzeit-Bedingung. Jede Task (und die enthaltenen

Bei dieser reinen Lastanalyse bleiben Integrations- und insbesondere Schedulingeffekte (mit Auswirkungen auf temporäre Spitzenlasten, Wirkketten-Latenzen oder Jitter) zunächst außen vor. Daher ist die Aussagefähigkeit bezüglich der Realisierbarkeit nur ein erster Richtwert.

Zyklen und Ausführungszeiten

Als nächstes wird das System nach Zykluszeiten der Teilfunktionen aufgeteilt. Diese sind aus den Software-Modellen, wie sie z. B. in Simulink oder ASCET existieren, bekannt. Beim Übergang in die Software-Architektur werden daraus die Runnables. Die Zykluszeit kann direkt in AUTOSAR-XML (ARXML) angegeben werden.

Ebenso lassen sich die Laufzeiten der einzelnen Runnables je nach Projektfortschritt aus den Erfahrungen früherer Projekte abschätzen, als Budgets in das Modell aufnehmen oder an Prototypen messen.

Bild 1 zeigt eine Software-Architektur (bzw. ein Funktions-Blockschaltbild) mit annotierten Zykluszeiten (ts) und Laufzeiten (et = execution time).

So entsteht Schritt für Schritt ein verfeinertes Lastmodell, mit dem sichtbar wird, welche Funktion bzw. welches Runnable wie viel Last „verbraucht“. Die Umrechnung lautet: Last = Laufzeit / Zykluszeit. Bei kritischen Lastsituationen kann man gezielte Maßnahmen ergreifen, z. B. Laufzeitbudgets einzelner Runnables oder Funktionen reduzieren oder Zykluszeiten erhöhen. Damit legt man verbindlich einen Rahmen

Runnables) muss in ihrem Zyklus komplett ausführbar sein, und zwar unter Berücksichtigung des Scheduling, insbesondere von Unterbrechungen durch höherprioritäre Tasks. Ansonsten kann das System zur Laufzeit Task-Aktivierung „verlieren“. Hierfür eignet sich das Schedulinganalyse-Tool SymTA/S. Idealerweise liegen die sogenannten Task-Antwortzeiten (inklusive Scheduling-Einfluss) zu Beginn einer ECU-Entwicklung noch mindestens 30 ... 50% unter der

Zykluszeit (Deadline). Auch hier wird die Reserve für spätere Integrationsstufen benötigt.

An dieser Stelle ist bereits die Basis für eine recht detaillierte Prüfung der Realisierbarkeit erreicht. Falls Deadlines sehr knapp erreicht oder gar überschritten werden, steigt das Risiko von späteren Timing-Fehlern signifikant. Durch das Verschieben von Runnables in langsamere Raster oder ggf. auf einen anderen Core, das Reduzieren von Laufzeitbudgets (entspricht einer Code-Optimierung) oder notfalls die Wahl einer leistungsstärkeren CPU lässt sich geeignet gegensteuern. Anhaltspunkte geben uns die Ergebnisse der Schedulinganalyse. Derartige Optimierungszyklen sind in SymTA/S leicht automatisierbar dank des virtuellen Task-Modells, der ARXML-Unterstützung und der effizienten Timing-Analysen. Sind alle Task-Antwortzeiten im erlaubten Bereich, können wir uns den Wirkketten zuwenden.

Wirkketten-Latenzen

Wirkketten entsprechen Signalpfaden entlang der Kommunikation zwischen Runnables (analog Signalwegen in den Simulink- oder ASCET-Blockdiagrammen), die funktionale „Wirkung“ wird dabei dadurch erzielt, dass ein Runnable Daten produziert, die vom nächsten Runnable in der Kette gelesen werden, z. B. von einem Sensor über mehrere Funktionen bis hin zu einem Aktuator. Für solche Wirkketten liegen in der Praxis oft Zeitbedingungen vor, z. B. erlaubte Reaktionszeiten oder Regler-Totzeiten, die es zu erfüllen gilt. Diese Ketten wie auch entsprechende Constraints können ab AUTOSAR 4.0 mit Hilfe des Timing-Template angegeben werden.

Zur Prüfung der Constraints wird nun die Latenz jeder Wirkkette bestimmt. Diese ergibt sich zum einen aus den Antwortzeiten derjenigen Tasks, die die Runnables der Wirkkette enthalten, zum anderen aus der durch die Funktionsentwicklung vorgegebenen „logischen“ Reihenfolge dieser Runnables. Hier kommt es häufig zu komplexen, nicht

immer leicht verständlichen Effekten. Die Zeit entlang einer Wirkkette wird in der Praxis dominiert von Unterbrechungen durch höherpriorige Tasks oder Interrupts, der Pufferung von Daten zwischen Runnables, Up-/Down-Sampling beim Wechsel der Zykluszeit etc. Die eigentlichen Runnable-Ausführungszeiten sind nur ein Baustein der Gesamtlatenz (siehe **Bild 2**).

Die SymTA/S Timing-Analysen lösen diese Abhängigkeiten auf und bestimmen automatisiert die gesuchten Wirkketten-Latenzen, die dann mit den Zeitvorgaben verglichen werden. Als Ergebnis sieht man, welche Wirkketten im System nahe an der Deadline oder darüber und somit grundsätzlich kritisch sind und ob erneut korrigierende Maßnahmen wie das Anpassen des Scheduling-Konzeptes, Zykluszeiten, Runnable-Mapping, usw. notwendig werden. Zeigen dann alle Ampeln auf „grün“, folgt der letzte Schritt.

ECU-Konfiguration

Hier muss man für den bislang nur grob vorkonfigurierten virtuellen Task-Schedule die Details der RTE- und OS-Konfiguration so festlegen, dass insbesondere die kritischen Wirkketten berücksichtigt werden. Zwei Parameter sind dabei von allergrößter Bedeutung.

Zum einen die bislang nicht betrachtete Runnable-Reihenfolge innerhalb der Tasks durch den Parameter „PositionInTask“: die wird immer dann elementar, wenn zwei in der Wirkkette aufeinanderfolgende Runnables im selben Task liegen. Dann nämlich sollte die PositionInTask entsprechend gewählt werden, weil man ansonsten eine ganze Task-Zykluszeit entlang der Wirkkette quasi verschenkt. Dies ist im Beispiel von *Bild 1* der Fall. Hier liest das Runnable R4 Daten von R8, welches wiederum erst nach R4 im 5-ms-Zyklus ausgeführt wird. Das bedeutet, R4 muss bis zum nächsten 5-ms-Zyklus warten.

Zum anderen sind die ebenfalls bisher ignorierten Task-Off-

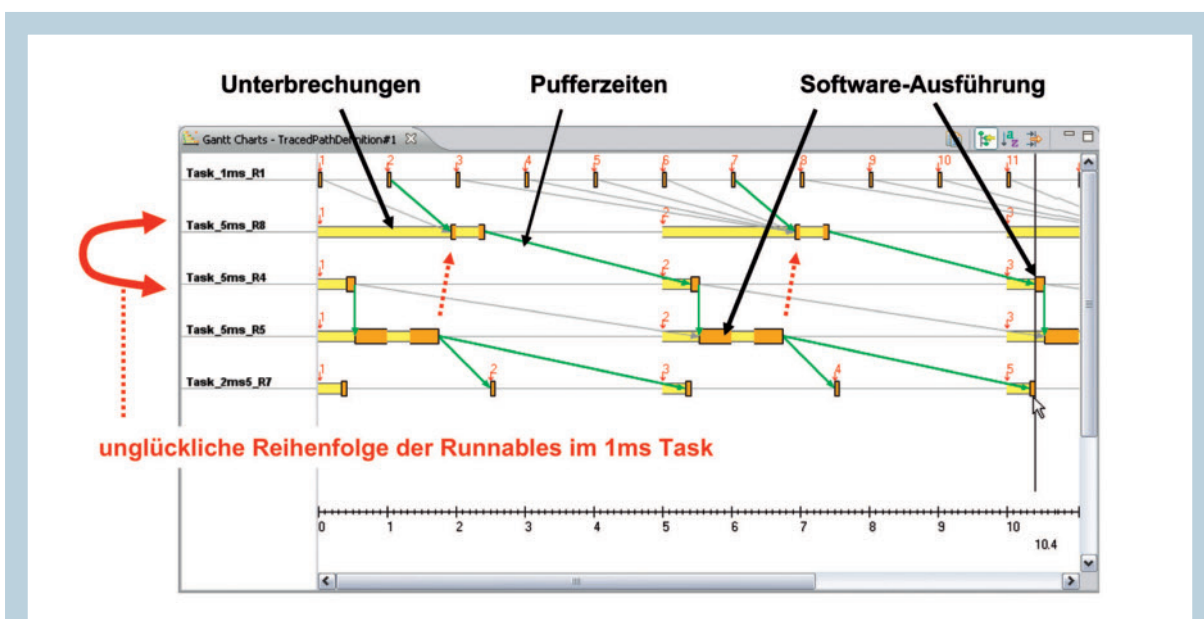


Bild 2: Wirkkette WK3 mit Scheduling-Effekten.

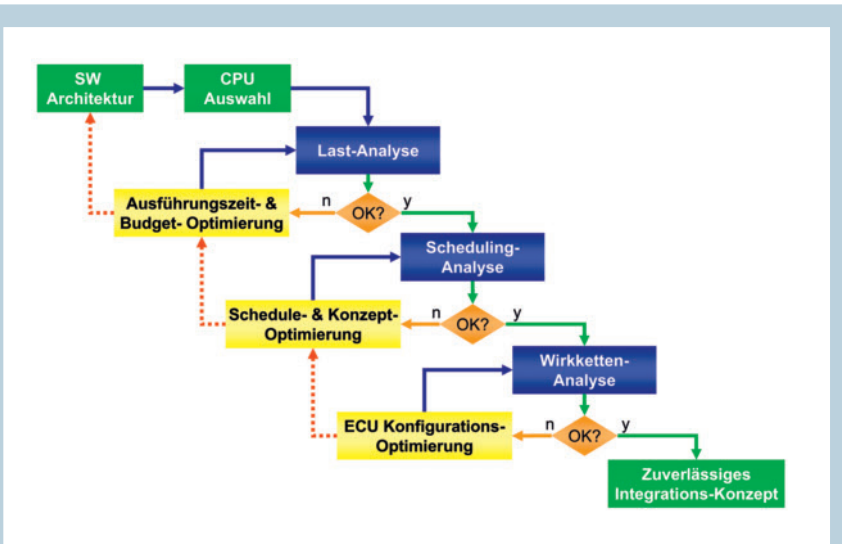


Bild 3: Ablauf der Methodik.

© automotive

sets wichtig: Mit Offsets lassen sich die Startzeitpunkte der Tasks gegeneinander verschieben und dadurch Lastspitzen reduzieren sowie die Wirkketten-Latenzen weiter optimieren. Bei Multi-Core-Systemen ist noch entscheidend, ob man gemeinsam (synchron) oder individuell sche-dulen möchte. Diese Werte sind in ARXML und SymTA/S verfügbar.

Schließlich erweitert man sein Timing-Modell noch um die „Zeitverbraucher“ aus der Basis-Software. Für diese wurden anfangs eine Last-Reserve vorgehalten, die sich jetzt konkretisieren lässt. Dabei konzentriert man sich auf diejenigen Teile, die als eigenständige Tasks (z. B. TX-COM) oder langlaufende Interrupts (z. B. RX-CAN oder auch Kurbelwelleninterrupts) den größten Einfluss auf den Schedule nehmen. So wird schließlich eine detaillierte Schedulinganalyse durchgeführt und erneut die Task-Antwortzeiten (zur Prüfung der Schedulability) und Wirkketten-Latenzen überprüft. Nach möglicherweise nochmals notwendigen Korrekturen ist die Systemkonfiguration dann im Wesentlichen abgeschlossen und die getroffenen Konfigurations-Entscheidungen (SW-Architektur, RTE/OS-Config, BSW) können mit minimalem Projektrisiko in die detaillierte Software-Entwicklung und Implementierung übernommen werden.

Durchgängigkeit

Die hier beschriebene und in **Bild 3** zusammengefasste Methodik ist durchgängig und sowohl für neue als auch für bereits existierende Systeme einsetzbar. Typische Projekte sind in der Regel heterogen; für wiederverwendbare Komponenten liegen detaillierte Timing-Daten vor, für neue muss die Laufzeit zunächst geschätzt werden, bis erste Messungen existieren. Zudem wächst die Software von einer Integrationsstufe zu nächsten und auch über Produktupdates, sodass immer wieder zu einem recht detaillierten Modell neue Teile hinzukommen, die dann in den bestehenden Schedule „virtuell“ integriert und schließlich final konfiguriert werden müssen. Das gilt auch für den Übergang zu Multi-Core-Systemen, wenn eine bekannte SW-

Architektur partitioniert wird.

Die vorgestellte Methodik, der AUTOSAR-Standard und das Werkzeug SymTA/S unterstützen solche heterogenen Timing-Modelle durchgängig über den gesamten Entwicklungszyklus, denn für das Timing-Modell ist es gleichgültig, ob schon Code vorliegt oder dieser bereits ausgemessen wurde (z. B. mit dem Symtvision TraceAnalyzer oder den Werkzeugen der Real-Time Experts). Für das Modell braucht man lediglich die Daten und kann Messungen mit Schätzungen und Budget kombinieren.

Das Ergebnis ist eine frühzeitige Optimierung und umfassende Bewertung der Realisierbarkeit auf Basis der aktuellen Informationen, die sich klar

darstellen, kommunizieren und argumentieren lässt, insbesondere gegenüber dem Projektmanagement, Software-Zulieferern und OEM-Kunden (Stichwort Lastenheft). Timing-Modelle, die nach AUTOSAR standardisiert und mit einem geeigneten Werkzeug wie SymTA/S automatisiert analysierbar sind, bilden ideale Entwicklungsartefakte zum Austausch zwischen verschiedenen Entwicklungsgruppen und entlang der Zulieferkette. Dabei ist die IP Protection garantiert, denn ein solches Timing-Modell enthält weder Code noch Algorithmik-Details. Mit dieser Methodik minimiert man also die Integrationsrisiken von ECU-Projekten a priori und hat die Echtzeiteigenschaften dauerhaft im Blick. So steht einer Kosten- und Ressourcenoptimierten Implementierung nichts mehr im Weg. (oe)

Im 2. Teil des Beitrags wird die Methodik an einem konkreten Beispiel angewendet und detaillierte Auswertungen geliefert.

Symtvision stellt seine Scheduling-Analyse-Toolchain auf der diesjährigen Embedded World vom 1. bis 3. März auf einem gemeinsamen Stand der Real-Time Experts aus (Halle 10, Stand 412).



Dr. Kai Richter ist anerkannter Experte im Bereich der Zeit- und Leistungsanalyse verteilter, eingebetteter Systeme. Seit 2005 ist er in der von ihm mitgegründeten Symtvision GmbH als Geschäftsführer für Technologie und Forschung verantwortlich.



Dr. Marek Jersak arbeitete als Ingenieur und Projektleiter für Conexant Systems im Bereich DSP Compiler-Design und Optimierung. Seit 2005 ist er Geschäftsführer (CEO) der Symtvision GmbH, die er mitgründete. Seine Aufgaben umfassen Unternehmensstrategie und Business Development.