MICRO CONSULT

Embedded-Software-Design: Anforderungen entwickeln und Architekturen verfeinern

“Predictive Maintenance” wird laut Experten in den nächsten fünf bis zehn Jahren bei so gut wie allen rotierenden Maschinen die Norm sein. Die rasante Entwicklung der Sensortechnologie und der künstlichen Intelligenz beschleunigt dieses Tempo zusätzlich. Die hierzu mit dem Internet of Things (IoT) verbundenen Embedded-Systeme stellen immer komplexere Anforderungen an das Design. Wer im Vorfeld durchdachte Anforderungen entwickelt und die Architektur konsequent mit Software-Design verfeinert, sichert Software- und Produktqualität.

Einordnung im Entwicklungsprozess

Im Entwicklungsprozess ordnet sich das Software-Design nach der Software-Anforderungs- und -Architekturentwicklung und vor der Implementierung ein. Die Entwicklung arbeitet heute inkrementell und iterativ und nicht wasserfallartig. Bildlich betrachtet repräsentiert Software-Design die Brücke zwischen der Architektur und der Implementierung.

Die Systemintelligenz bei IoT- und Industrie-4.0-Produkten liegt sehr häufig in der Embedded-Software. Deren Komplexität steigt aktuell stark an. Viele Unternehmen haben inzwischen die Notwendigkeit der Embedded-Software-Anforderungserfassung und der Embedded-Software-Architekturentwicklung erkannt. Die logische Konsequenz: mithilfe von Softwaredesign das Ergebnis des Softwarearchitekten vor der Implementierung verfeinern.

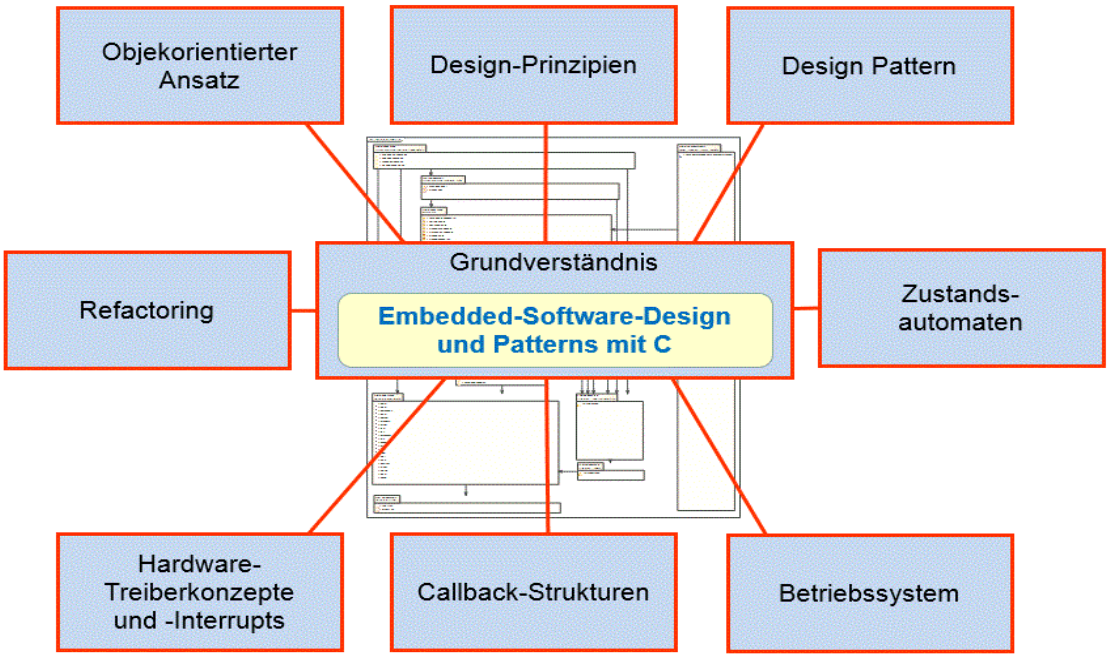

Abb. 1: Themen aus dem Embedded-Software-Design

Unabhängig von C++ mit objektorientiertem Ansatz

Objektorientierte Software unterstützt positive Software-Qualitätsmerkmale wie Erweiterbarkeit, Wartbarkeit, Portierbarkeit, Wiederverwendbarkeit und vieles mehr. Bei der objektorientieren Softwareentwicklung können Sie außerhalb von C++ Konstrukte wie Klassen, Objekte, Assoziationen, Aggregation, Komposition, Vererbung sowie polymorphe Interfaces auch in C programmieren.

Design-Prinzipien helfen beim Clean Code

Nicht zuletzt durch die in den letzten Jahren immer populärer gewordene „Clean Code Developer"- Initiative ist das Thema „qualitativ hochwertige Software entwickeln" nach wie vor aktuell. Die Einhaltung einfacher Prinzipien wie DRY, KISS, SLA, SRP, POLS hilft Entwicklern dabei, Software von hoher Qualität zu schreiben.

Schneller entwickeln mit Design Patterns

Design Patterns sind praxisbewährte Lösungen zu immer wiederkehrenden Herausforderungen in der Softwarenentwicklung. Je mehr Design Patterns Sie kennen, desto schneller geht die Softwareentwicklung von der Hand. Design Patterns lassen sich in Erzeugungs-, Struktur- und Verhaltenspatterns unterteilen.

Design Patterns speziell für Embedded-Software lassen sich in die folgenden Themen gruppieren: funktionale Sicherheit, Zugriffssicherheit, Zuverlässigkeit, Hardwarezugriff, Parallelisierung, Ressourcenzugriff und Multicore.

Automatenfunktionalität von der Applikation trennen

Das generische (für alle Fälle gültige) Verhalten in Embedded-Software ist sehr häufig durch Zustandsautomaten zu designen und zu implementieren. Doch wie modelliere (dokumentiere) und wie implementiere ich den Zustandsautomaten?

Die spezielle Herausforderung bei der Implementierung in C besteht darin, die Automatenfunktionalität von der Applikation strikt zu trennen. Dieser Ansatz verbessert die Software-Qualitätsmerkmale erheblich.

Betriebssystem verpflichtend

Mit komplexerer Software ergibt sich häufig auch die Möglichkeit des sinnvollen (quasi-) parallelen Abarbeitens von Aufgaben innerhalb der Software. Dazu setzen Sie im Design Mechanismen wie Multithreading, Synchronisation und Kommunikation aus einem (Embedded-/Echtzeit-) Betriebssystem mit ein.

Bei der Entwicklung einer IoT- (Internet of Things) Applikation sind Sie meist durch die eingebundenen Kommunikationsstacks dazu gezwungen, ein Betriebssystem zu verwenden.

Callback-Strukturen als Mittel der Wahl

Um zwischen Softwarearchitektur-Elementen wie beispielsweise Softwareschichten entkoppelt zu kommunizieren, sind Callback-Strukturen häufig das Mittel der Wahl. Hierbei gibt es verschiedene Design- und Implementierungsansätze, ob mit oder ohne Betriebssystem, objektorientiert oder prozedural und mit oder ohne Polymorphie (statisch oder dynamisch).

Hardware-Treiberkonzepte und -Interrupts

Falls Sie die durch den Siliziumhersteller gelieferten oder durch Tools generierten Treiber nicht verwenden (wollen oder dürfen) und auch Ihr Betriebssystem kein I/O-Management bereitstellt, haben Sie auch die Möglichkeit, ein Treiber- und Interrupt-Service-Konzept zu entwickeln, welches Ihre Software-Anforderungen bestens erfüllt.

Refactoring

Nicht jeder Entwickler hat die Möglichkeit, "auf der grünen Wiese" mit einer komplett neuen Embedded-Softwareentwicklung zu starten. Oft geht es darum, bestehende Software zu warten und zu erweitern, die mitunter teilweise länger als zehn Jahre gereift und gewachsen ist. Das macht gravierende Veränderungen schwierig, da dabei immer wieder Nebeneffekte auftreten.

Entscheiden Sie sich dann gegen den Neubeginn, muss die Software schrittweise umstrukturiert werden, damit die Nebeneffekte wieder verschwinden oder erst gar nicht auftauchen. Umstrukturieren bedeutet, dass Sie in geplanten Projekten im Kleinen (bei Daten, Funktionen, Klassen und Objekten) optimieren und unter Umständen auch im Großen die Softwarearchitektur anpassen.

Fazit

Jeder Embedded-Softwareentwickler muss sich der hohen Relevanz des Themas Software-Design bewusst sein. Das Wissen darüber führt zur unmittelbaren Verbesserung der Software- und damit der Produktqualität.

Das Know-how qualifiziert Sie als Embedded-Software-Entwickler für den nächsten Schritt in Ihrer beruflichen Karriere. Nutzen Sie Ihre Möglichkeit dazu und holen Sie sich das erforderliche Fachwissen im MicroConsult-Seminar „Embedded-Software-Design und Patterns mit C".

MICROCONSULT

Autor: Thomas Batt

Thomas Batt studierte nach seiner Ausbildung zum Radio- und Fernsehtechniker Nachrichtentechnik. Seit 1994 arbeitet er kontinuierlich in verschiedenen Branchen und Rollen im Bereich Embedded-/Realtime-Systementwicklung. 1999 wechselte Thomas Batt zur MicroConsult GmbH. Dort verantwortet er heute als zertifizierter Trainer und Coach die Themenbereiche Systems/ Software Engineering für Embedded-/Realtime-Systeme sowie Entwicklungsprozess-Beratung.

Mehr Information

MicroConsult Training & Coaching zu Embedded- und Echtzeitprogrammierung

MicroConsult Training & Coaching zum Thema Softwarequalität

MicroConsult Fachwissen Embedded- und Echtzeit-Softwareentwicklung