

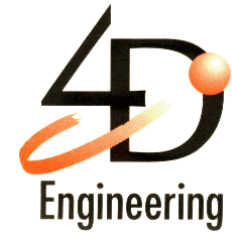
Rapid Embedded Software Development

mit wiederverwendbaren Softwarekomponenten

Matthias Bauer, 4D Engineering
bauer@four-d.de
<http://www.four-d.de>

ESE-Kongress, 06.12.2011

Software-Wiederverwendung in der Praxis



- Software-Wiederverwendung ist in vielen Bereichen seit geraumer Zeit etabliert (GUI-Applikationen)
- Für Embedded-Software wird dagegen wiederkehrende Funktionalität oft neu erfunden
- Behauptung: Auch bei Embedded-Software gibt es erhebliches Potential wiederverwendbare Komponenten einzusetzen

Definition: Wiederverwendbare Komponenten

Was ist **nicht** gemeint?

- Per Copy und Paste vervielfältigter Code
- Protokollstacks, Betriebssysteme, usw.

Was ist gemeint?

- Kleine parametrierbare Einheiten, die genau einen Aspekt / eine Funktionalität implementieren
- Generisch einsetzbare Komponenten (Baukasten)
- Anwendungs- oder plattformspezifische Komponenten (z. B. Algorithmen, Applikationsfunktionalität, Treiber)

Historie

Anforderungen an Embedded Software früher:

- Überschaubare Funktionalität und Applikationskomplexität
 - Hauptaufgabe: Treiberentwicklung
 - Potential und Motivation zur Wiederverwendung hardwareunabhängiger Teile war gering
- Extrem begrenzte Ressourcen zwangen zur „maßgeschneiderten“ Implementierung

Anforderungen heute

- Embedded-Plattformen bieten heute mehr Ressourcen (Preisverfall)
- Umfang und Komplexität der Anforderungen an Embedded-Anwendungen steigen (Konfigurationsschnittstellen, In-Target-Software-Update, ...)
- Sämtliche Funktionalität neu zu implementieren wird zunehmend schwieriger, wegen
 - Zeit- und Kostendruck
 - Qualitätsanforderungen

Problematik

- Vergleichsweise wenig Erfahrung zur Abstraktion wiederwendbarer Funktionalität vorhanden
- Identifikation wiederverwendbarer Komponenten ist schwieriger als für z. B. GUI-Anwendungen
- Verfügbare Ressourcen sind begrenzt
- Psychologische Vorbehalte gegen den Einsatz von C++

Vorteile durch den Einsatz wiederverwendbarer Komponenten



Generische Komponenten:

- Zeit- und Kostenersparnis („Time to market“)
- Höhere Qualität

Spezifische Komponenten:

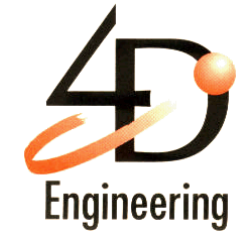
- Bessere Testbarkeit => höhere Qualität
- Einfaches Variantenmanagement
- Portierbarkeit auf andere Plattformen

Besonderheiten von Embedded-Komponenten

- Compiler unterstützen den Sprachumfang unvollständig oder nur ineffizient (z. B. Exceptions)
- Sparsamer Umgang mit Ressourcen nötig, z. B.
 - dynamische Speichieranforderung
 - Verwendung der kleinstmöglichen skalaren Datentypen (char, short, long, ...)
 - dosierter Einsatz von virtuellen Methoden (durch die Option zur Verwendung statischer Polymorphie)

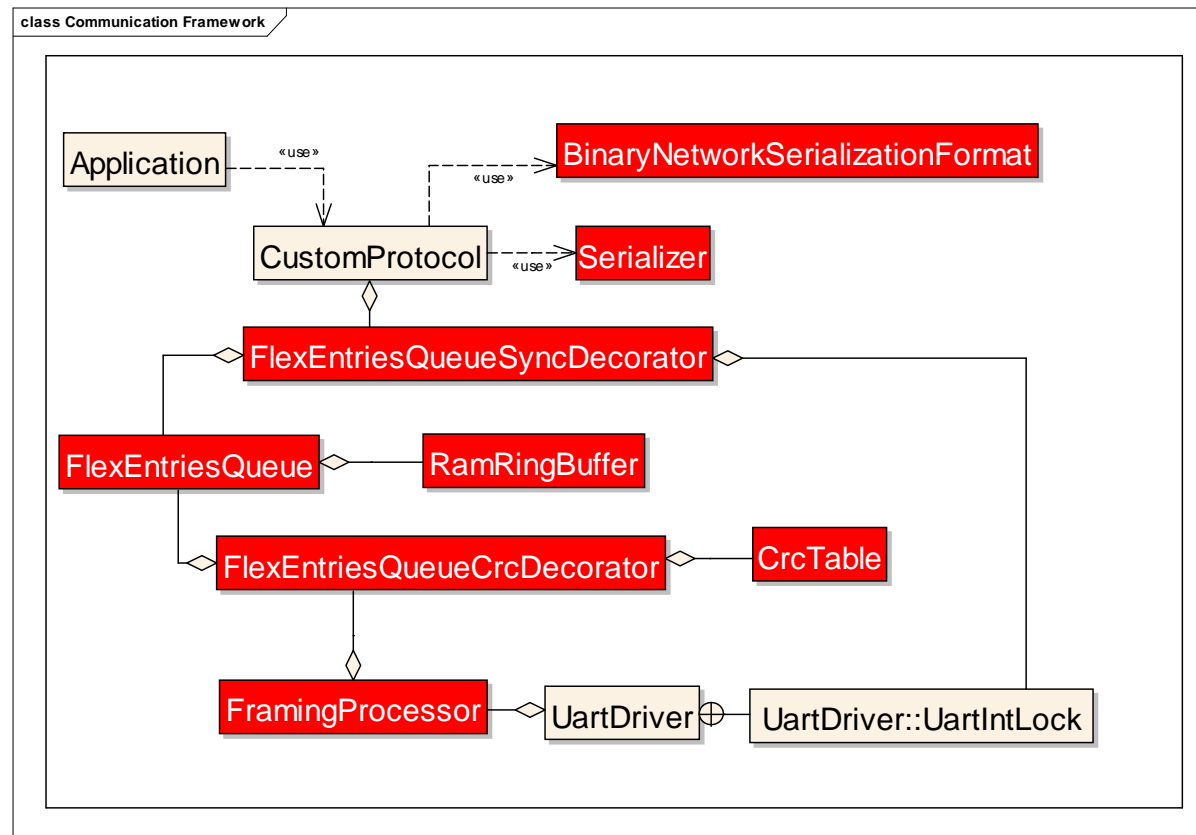
- ▶ Folgerung: Komponenten der STL, Boost, usw. sind oft nur bedingt geeignet

Beispiele für generische, wiederverwendbare Komponenten



- Containerklassen (Ringpuffer, Listen, Vektoren, Memory-Pools, ...)
- Debug-Logging (Formatierung, Konfiguration, ...)
- In-Target-Software-Update (Workflow, Prüfsummenberechnung, Fehlerbehandlung, Image-Auswahl im Bootloader, Anwendungsprotokoll)
- Tasküberwachung
- Kommunikationsprotokolle (Framing, Prüfsummenberechnung, (De-)Serialisierung)
- Konfigurationsdatenverwaltung in Flash-Memory
- Domänenspezifische Software-Komponenten (z. B. Regelungstechnik)

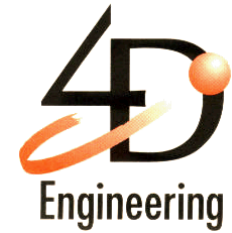
Beispiel: redBlocks Communication-Framework



Glue-Logic – Komponenten verbinden

- Die Komponenten müssen miteinander bekannt gemacht werden
- Das kann entweder zur Compilezeit oder bei der Initialisierung erfolgen
- Durch den gezielten Einsatz von statischer und dynamischer Polymorphie kann dabei hinsichtlich der benötigten Ressourcen optimiert werden.

Schlüssel zur Verwendung wiederverwendbarer Softwarekomponenten



- Baukasten mit generischen Komponenten
- Komponentensammlung (domänenspezifisch)
- Ausbildung / Know-How
 - C++-Sprachmittel (Polymorphie, Templates)
 - Embedded-Erfahrung mit C++
 - Intensiven Überblick über die verfügbaren Komponenten
- Umfeld

Zusammenfassung

- Der Einsatz wiederverwendbarer Komponenten in Deeply Embedded Anwendungen wirkt sich positiv aus, hinsichtlich
 - Zeit
 - Kosten
 - Qualität
 - Wartbarkeit
- Allerdings müssen die Komponenten auf diesen Einsatzzweck zugeschnitten sein.
- Nötige Investition: Ausbildung der Mitarbeiter

Rapid Embedded Software Development mit wiederverwendbaren Softwarekomponenten

Matthias Bauer, 4D Engineering
bauer@four-d.de
<http://www.four-d.de>

ESE-Kongress, 06.12.2011