

Framework für Embedded SW

Framework zur Gliederung und
Kapselung von Embedded Software
unter Verwendung des OO-Ansatzes

Hugo Ziegler

Agenda

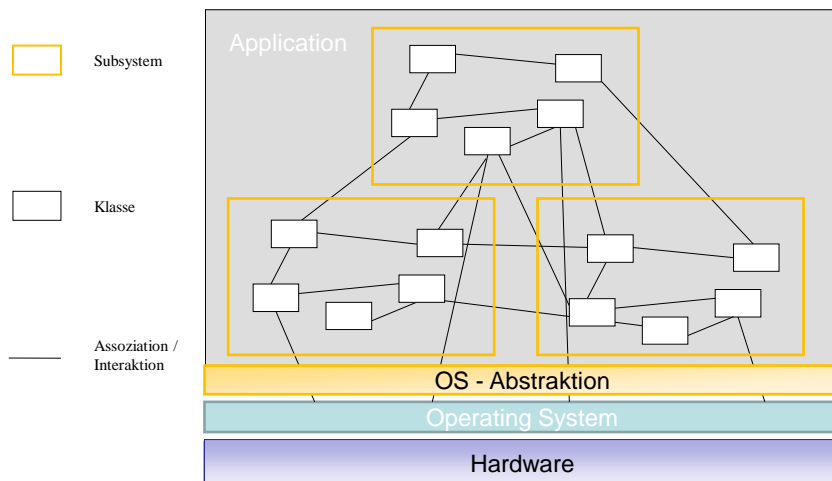
- Motivation ein eigenes Framework zu «bauen»
- Übersicht über die Problemstellung
- Strategien und Konzepte
- Multi-Tasking
- Testen / Unit-Test

Motivation

Ein Framework soll uns unterstützen indem es...

- ...für «alltägliche Aufgaben» eine vordefinierte Implementation bietet.
- ...eine allgemein verbindliche Vorlage für den Bau von Teilsystemen liefert. (Lego System)
- ...durch geeignete «Layers» die Wiederverwendung ganzer Software-(Sub)systeme entscheidend vereinfacht.
- ...den Bau hierarchischer Software ermöglicht.

Übersicht



Strategien und Grundregeln

Definition Building Block:

- Ein Building Block ist ein Software-Subsystem, das aus einer Gruppe von Klassen besteht, welche eine definierte Aufgabe erfüllen.
- Der Building Block definiert eine generische Architektur für ein solches SW-Subsystem
- Ein Building Block erscheint gegenüber der Applikation als eine einzige Klasse.

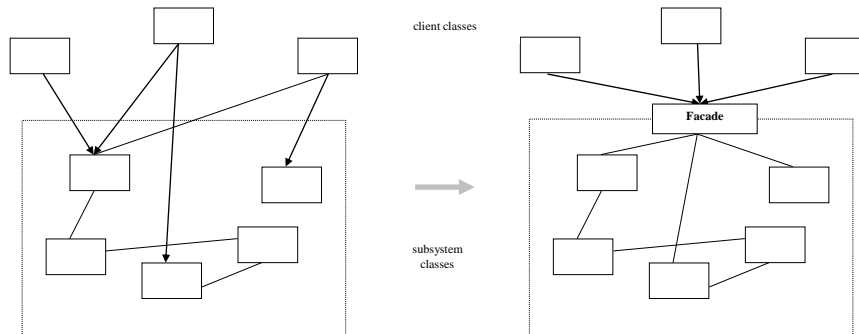
Strategien und Grundregeln (2)

Damit ein Building Block wiederverwendbar wird, braucht es Richtlinien für die folgenden Aspekte:

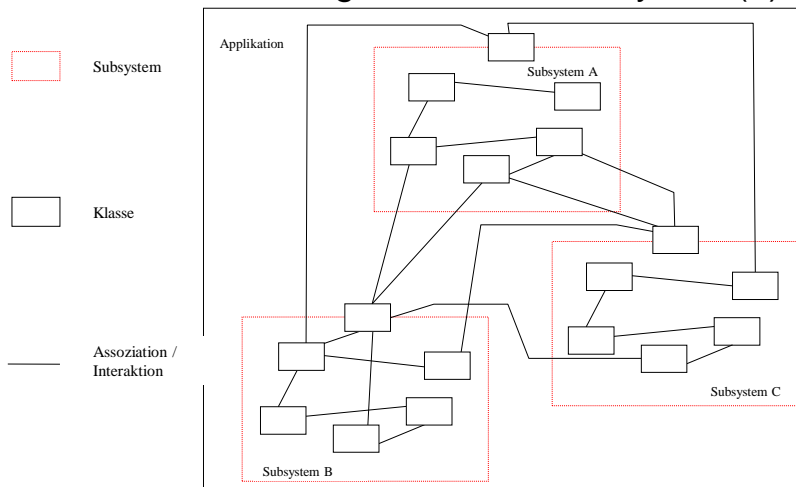
- Schnittstellen
- Lebenszeit – Erzeugung und Löschung von Subsystemen
- Portierbarkeit
- Parallelität und Konkurrenz

Schritt 1 – Zugriffe auf ein Subsystem

„Die Fassade definiert eine übergelagerte Schnittstelle, die das Verwenden des Subsystem einfacher macht“

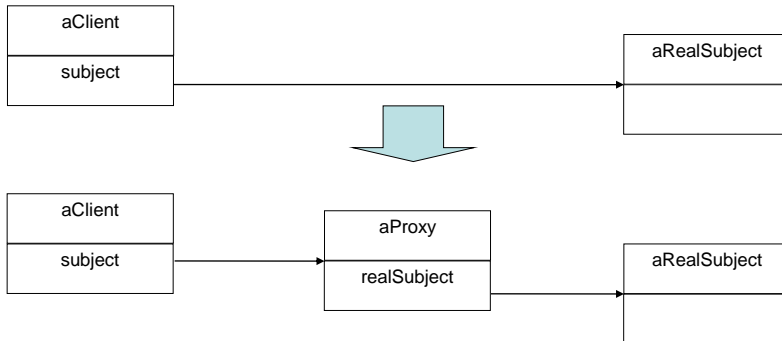


Schritt 1 – Zugriffe auf ein Subsystem(2)

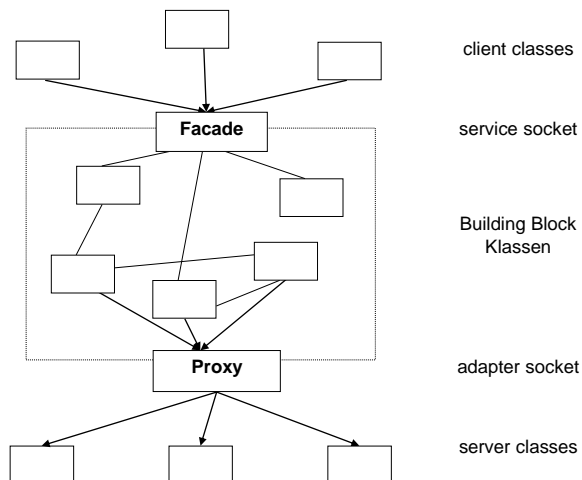


Schritt 2 – Zugriffe nach Aussen

„Stellen Sie einen Stellvertreter - oder Platzhalter - für einen anderen Gegenstand zur Verfügung, um so den Zugriff auf ihn zu regeln!“



Schritt 2 – Zugriffe nach Aussen (2)



Namen und Definition der Sockets

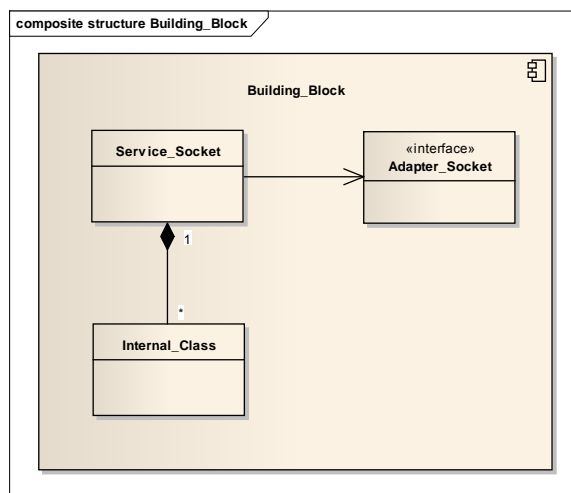
Definition: Service Socket

- Der **Service Socket** ist die Schnittstelle in den Building Block. Er repräsentiert alle Funktionalität, welche der Building Block zur Verfügung stellt, und er verbirgt die Details der internen Implementierung.

Definition: Adapter Socket

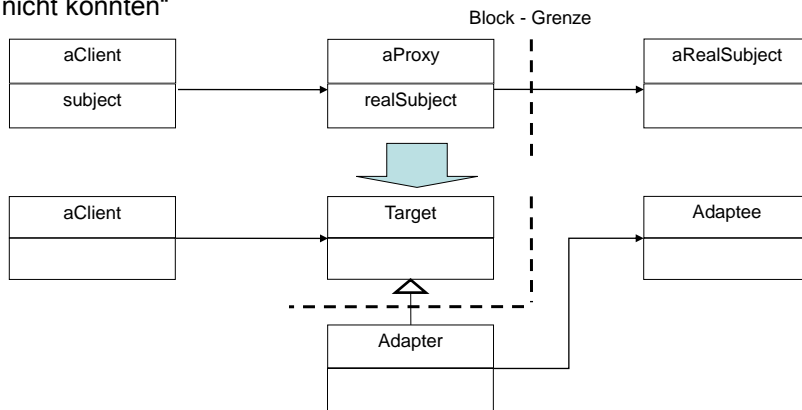
- Der **Adapter Socket** regelt alle Beziehungen der Building Block Mitglieder zur "Aussenwelt".
- Die Implementierung des Adapter Socket ist applikations-spezifisch.
- Die Spezifikation (das Interface) des Adapter Socket wird jedoch vom Building Block vorgegeben.

Schritt 3 – Lebensdauer

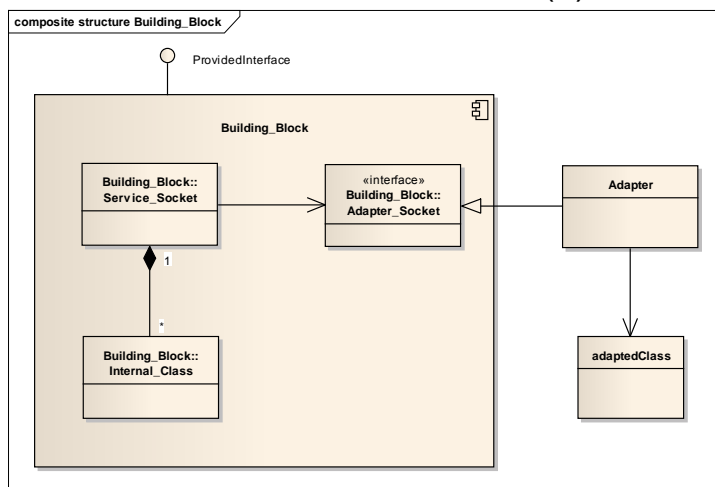


Schritt 4– Portierbarkeit

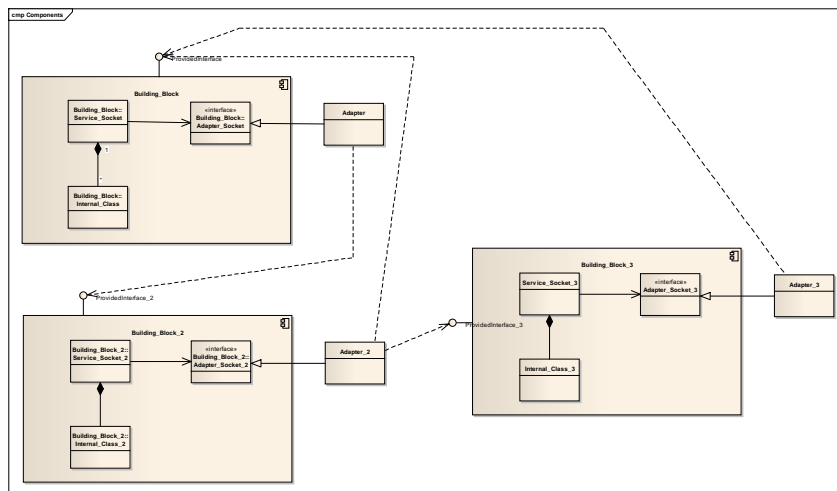
- Wandeln Sie die Schnittstellen einer Klasse in solche um, die von den Schnittstellen der Klienten erwartet werden. Der Adapter lässt Klassen zusammenarbeiten, die dies wegen der inkompatiblen Schnittstellen nicht könnten“



Schritt 4 – Portierbarkeit (2)



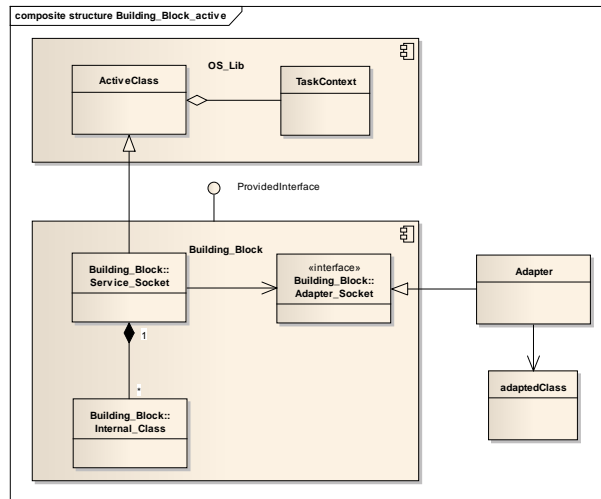
Neue Struktur einer Applikation



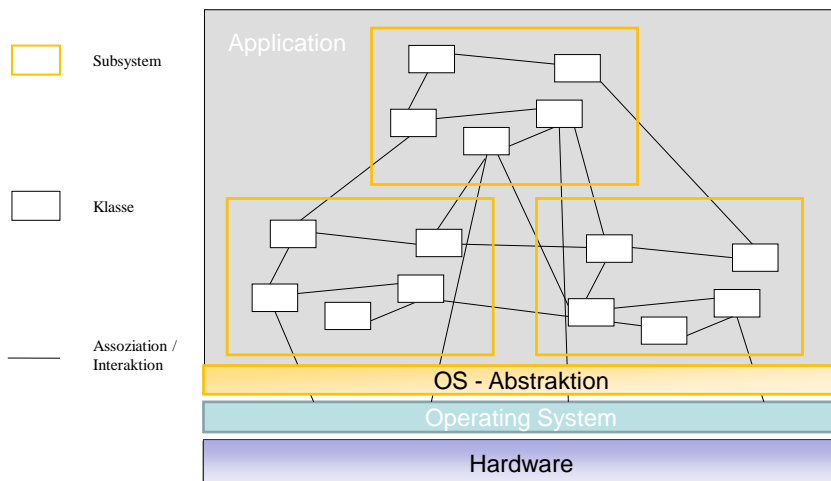
Multitasking und Konkurrenz

- Ein «Building Block» hat keinen oder einen Task (Ausnahmen sind in gut begründeten Fällen zulässig).
- Im Falle, dass ein «Building Block» einen eigenen Task Kontext hat, so sind alle Kontext relevanten Dinge in der Klasse «Kontext» behandelt.
- Alle Funktionen, die für eine Kontext Umschaltung benötigt werden, sind von der Basisklasse "Active_Class" geerbt.
- «Building Blocks» welche keinen eigenen Task Kontext besitzen, sind nicht von der Basisklasse "Active_Class" abgeleitet.

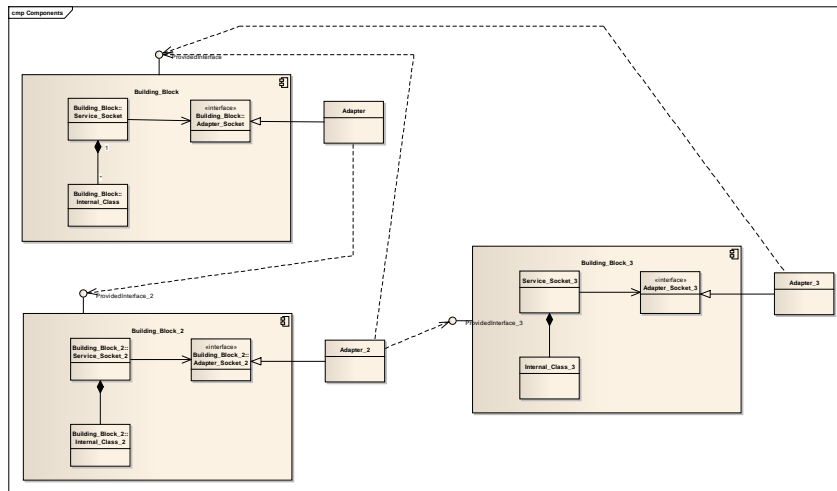
Multitasking und Konkurrenz (2)



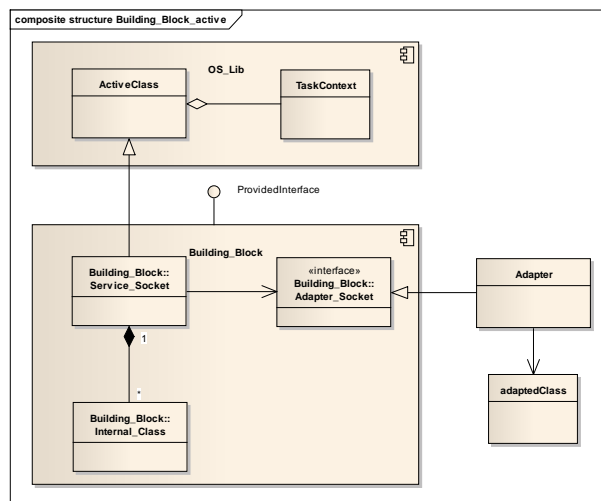
Rückblick – Ursprüngliche Struktur der Applikation



Rückblick – Neue Struktur einer Applikation



Rückblick – Abstraktion des OS



Testen

- Building-Blocks sind für Blackbox-Unittests bestens vorbereitet:
 - Ein Building-Block erscheint als eine Klasse (Service-Interface)
 - Die Zugriffe nach Aussen sind durch eine abstrakte Klasse geregelt (Adapter)
 - Durch die Kapselung lässt sich der Building-Block in der Testumgebung instanzieren.

- Mit den Libraries “Google Test” und “Google Mock” stehen zwei leistungsfähige Opensource Frameworks zur Verfügung, die im Gespann das Testen von Building-Blocks einfach machen

Testen (2)

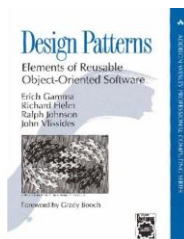
- Vorgehen
 - Der Building-Block wird in einem “Test-Fixture” mit allen nötigen Elementen aus dem Framework instanziiert und initialisiert.
 - Ein “Mock” für den Adapter wird vorbereitet.
 - Nun können die einzelnen Test für die verschiedenen Service-Methoden geschrieben und Erwartungen für die Aufrufe im Adapter formuliert werden. Dabei wird das Test-Fixture und der Adapter-Mock bei jedem Test neu erstellt.

- Hinweis
 - Getestet wird jeweils nur ein Block (der Building-Block). Das restliche Framework darf verwendet werden. Dessen Funktionalität wird in anderen Tests sichergestellt.

Screenshot eines Unittests

```
Running main() from gmock_main.cc
[====] Running 3 tests from 1 test case.
[-----] Global test environment set-up.
[-----] 3 tests from MMIServiceTest
[ RUN   ] MMIServiceTest.empty
[      OK ] MMIServiceTest.empty (0 ms)
[ RUN   ] MMIServiceTest.NotifyButtonDown
[      OK ] MMIServiceTest.NotifyButtonDown (0 ms)
[ RUN   ] MMIServiceTest.GetSliderState
[      OK ] MMIServiceTest.GetSliderState (0 ms)
[-----] 3 tests from MMIServiceTest (10 ms total)

[-----] Global test environment tear-down
[====] 3 tests from 1 test case ran. (10 ms total)
[ PASSED ] 3 tests.
```



Quellennachweis

- Design Patterns:
Gamma/Helm/Johnson/Vlissides, Design Patterns - Elements of Reusable Object-Oriented Software, 1997
- Google Test:
<http://code.google.com/p/googletest/>
- Google Mock:
<http://code.google.com/p/googlemock/>



Vielen Dank für die Aufmerksamkeit



Kontakt

Interessiert an unseren Dienstleistungen? Gerne stellen wir Ihnen in einem persönlichen Gespräch CSA Engineering AG näher vor.

CSA Engineering AG
Hans Huber-Strasse 38
CH-4500 Solothurn

Fon: +41 (0)32 626 35 55
Fax: +41 (0)32 626 35 50
www.csa.ch
info@csa.ch