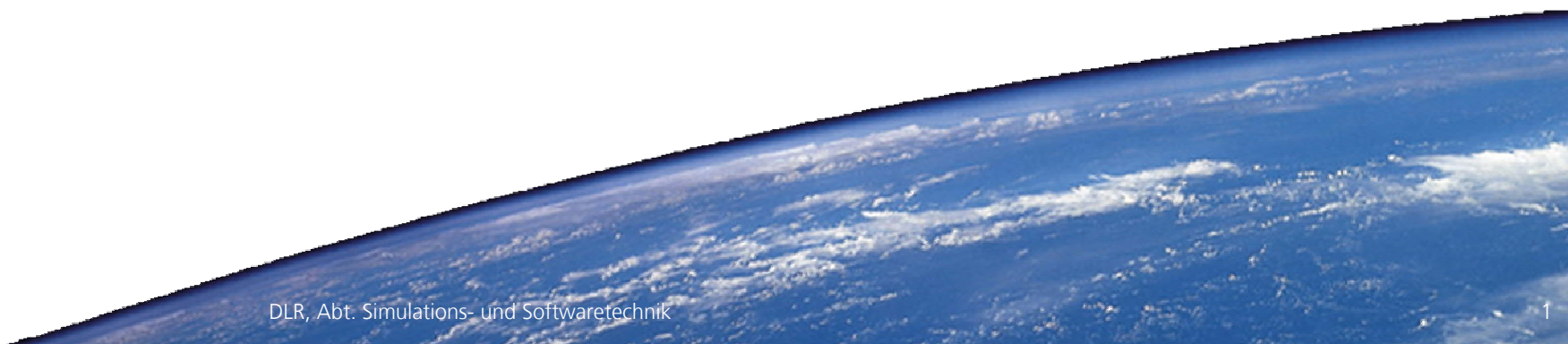
Einsatz von UML und C++ am Beispiel einer Satelliten-Lageregelungssoftware

Dipl. Inform. Olaf Maibaum
DLR, Abt. Simulations- und Softwaretechnik



Übersicht

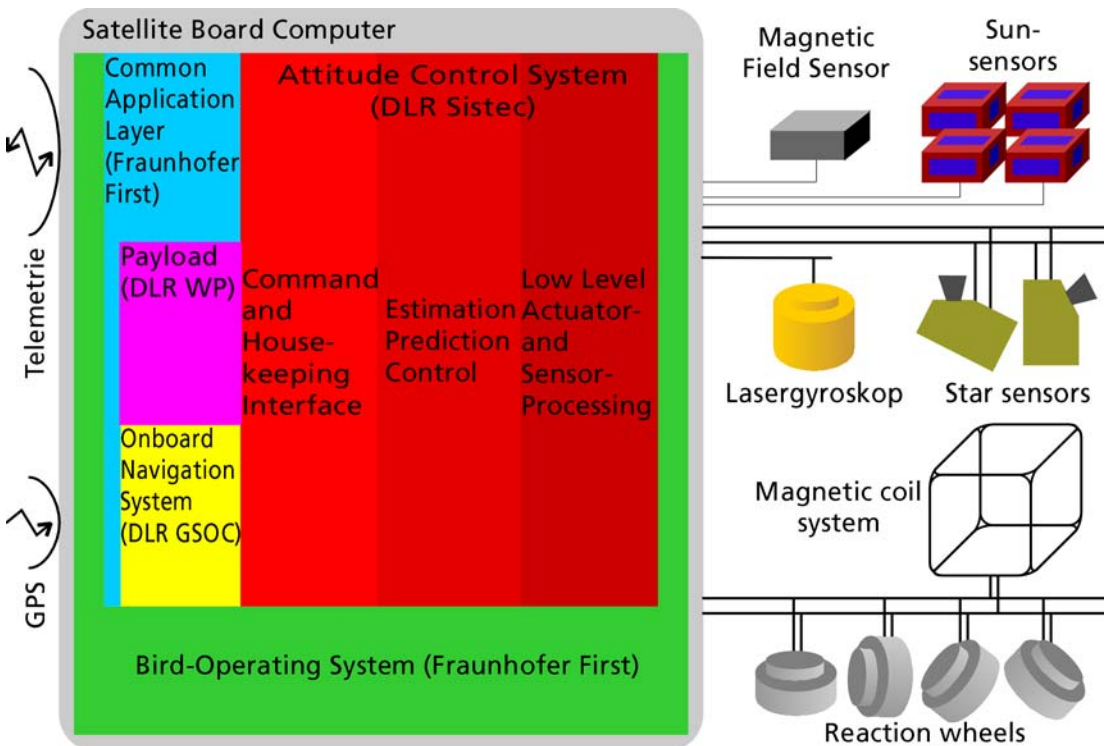
- Bird-Satellit und Einbindung der Lageregelung
- Zielplattform und Entwicklungsumgebung
- Designregeln für die Software
- Einblicke in das Lageregelungssystem
- Schwächen der UML
- Stärken des objektorientierten Ansatzes

BIRD

Ziele

- Weltraumerprobung einer neuen Generation von IR-Array-Sensor-Systemen, die die IR-Fernerkundung der Erde mittels Kleinsatelliten ermöglicht
- Detektierung und wissenschaftliche Untersuchung von Hochtemperatur-ereignissen
- Entwicklung und Demonstration von Kleinsatellitentechnologien

Umgebung der Lageregelung



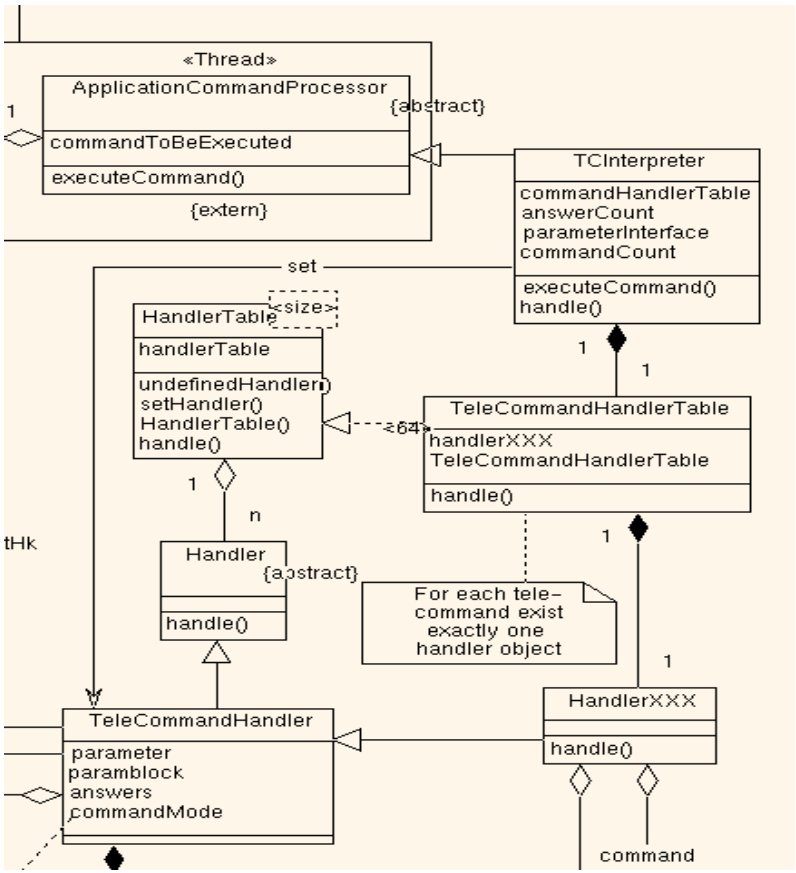
Zielplattform und Entwicklungssystem

- Zielplattform:
  - PowerPC-Board entwickelt von GMD-First mit einem warm redundanten Board, und zwei Board´s in kalter Redundanz
  - Objektorientiertes Betriebssystem von GMD-First
- Entwicklungsplattform:
  - Linux-PC
  - gcc-Compiler
  - doc++ zur Dokumentation des Detailed Designs
  - STP von Aeonix auf Solaris-Server als UML-Werkzeug
  - CVS auf Linux-Server zum Software-Konfigurationsmanagement
- Modul-Test oberhalb der Low-Level-Ebene auch auf Linux-PC möglich

Designregeln

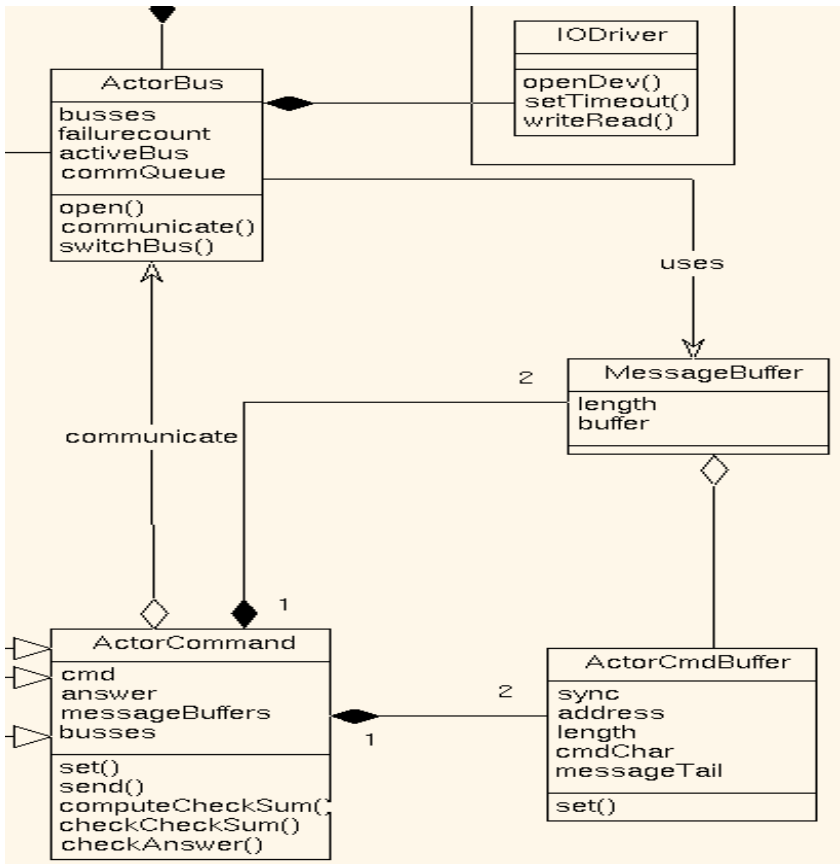
- Keine Verwendung von `new` und `delete` zur dynamischen Erzeugung von Objekten
- Keine Verwendung von `throw`, `try` und `catch`
- Alle Systembestandteile müssen Instanzen der Applikationsklasse sein, welche durch den Common Application Layer zur Verfügung gestellt wird
- Keine globalen Variablen verwenden
- Konstruktoren von abstrakten Datentypen sollten den Datentyp nur in einen definierten Zustand bringen, Destruktoren sind in der Regel leer
- Dynamische Strukturen bedienen sich der Betriebssystemklasse `Pool`, welche eine definierte Anzahl von Objekte eines definierten Typs bereit stellt
- Verknüpfungen zwischen Objekten werden nur in der Initialisierungsphase hergestellt, nicht im Nominalbetrieb des Systems
- Größere Datentypen sind bei der Parameterübergabe vom Typ „Call by Reference“
- Methodenrümpfe sind kurz

Telecommand-Schnittstelle



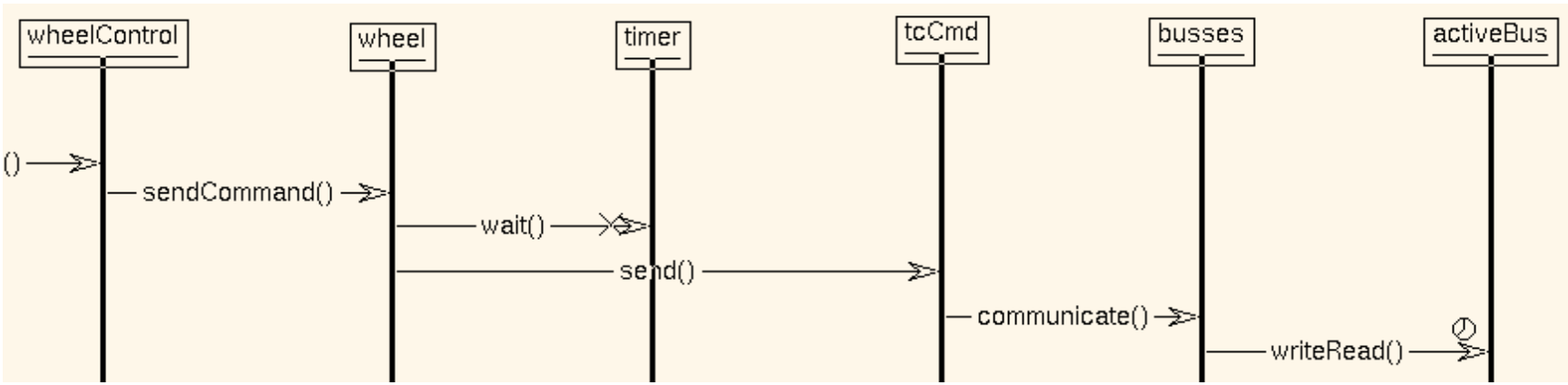
- Basisklasse für den Kommandointerpreter wird vom CAL bereitgestellt
- Zeiger auf Instanzen der Handlerobjekte werden in einer Handlertabelle bereitgestellt
- Aus dem Kommandocode und der Applikations-ID wird das Handlerobjekt in der Handlertabelle ermittelt und `handle` ausgeführt
- Vorgehen ist laufzeiteffizient
- Im Gegensatz zu `switch` kann kein `break` vergessen werden
- Einfaches Hinzufügen weiterer Kommandos ohne Seiteneffekte

Redundantes Bussystems



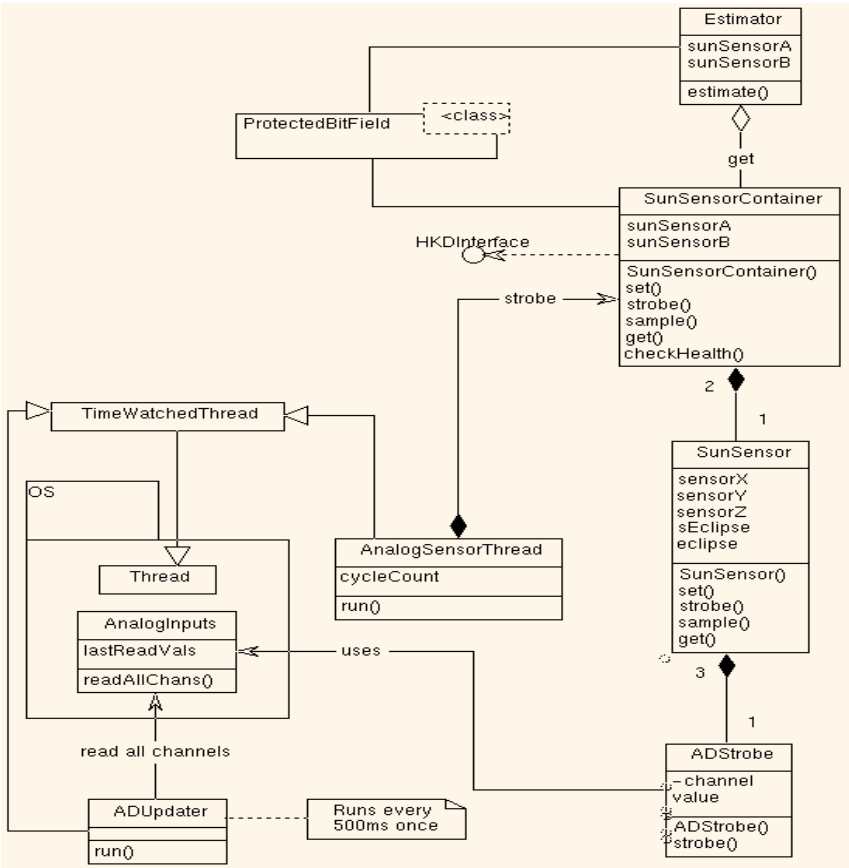
- Kommunikation mit intelligenten Geräten über redundanten Bus
- Betriebssystem stellt nur einen einzelnen Bus zur Verfügung
- Fehlerprotokoll für das redundante Bussystem ist in eine Klasse gekapselt
- Von `ActorCommand` abgeleitete Klassen kapseln den kommando-spezifischen Zugriff auf das Feld `messageTail` des `ActorCmdBuffer`
- Funktionalitäten die allen Kommandos gemeinsam sind, werden durch `ActorCommand` bereitgestellt

Ablauf der Wheel-Kommunikation



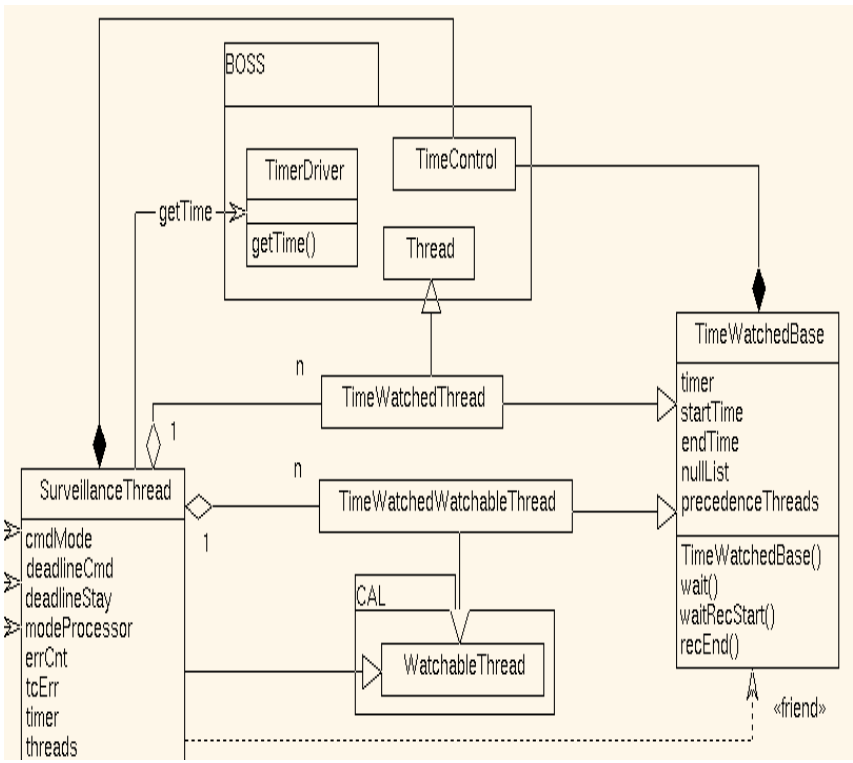
- Sequenzdiagramme sind problematisch in der Anwendung, da
  - sie schnell unübersichtlich werden
  - für jede Kombination von Spezialfällen ist ein Sequenzdiagramm notwendig
- Das gleiche gilt für Kollaborationsdiagramme

Zugriff auf AD-Wandler am Beispiel Sonnensensor



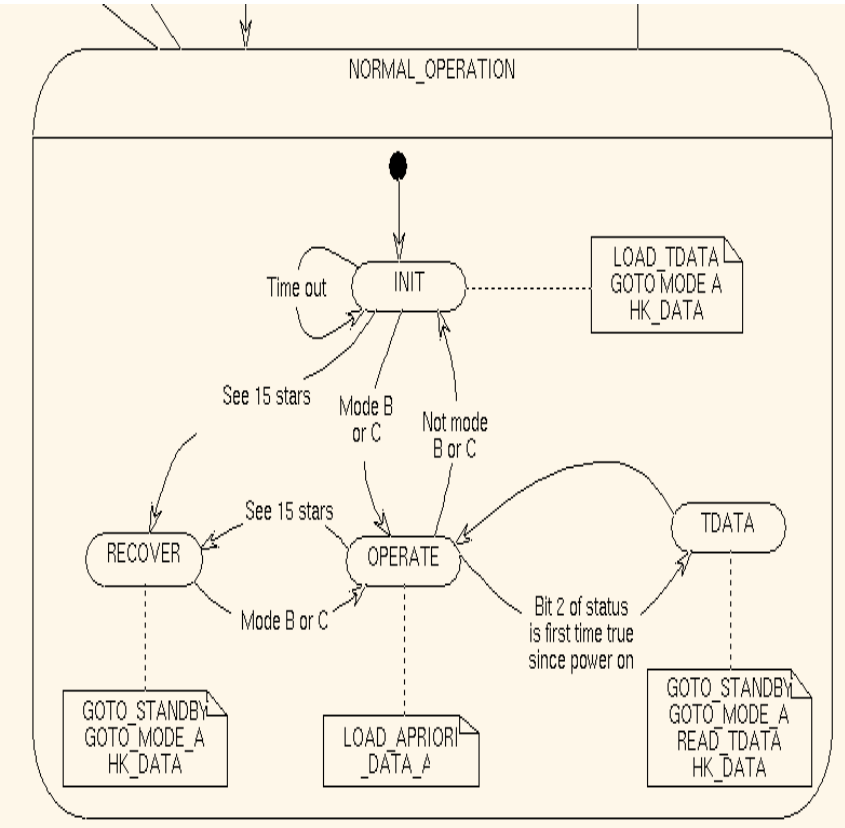
- Zugriff auf gelesene Analogwerte werden durch Klasse gekapselt, welche eine Konvertierung in den physikalischen Wert und eine Bereichsüberprüfung vornimmt
- Klasse `ADStrobe` bietet Interface für SILS der physikalischen Umwelt
- Die Erfassung der Analogwerte und die Konvertierung von Analogwerte in physikalische Werte werden von unterschiedlichen Leichtgewichts-prozessen ausgeführt, dadurch zeitliche Entlastung zum Zeitpunkt 0 eines Regelzyklus

Zeitüberwachung von Leichtgewichtsprozessen



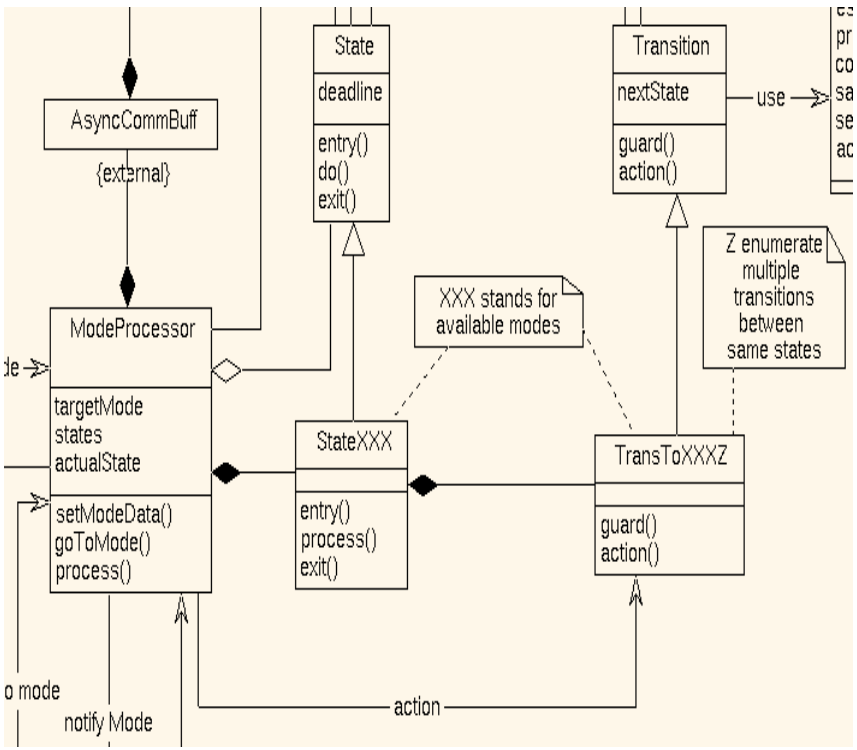
- Alle 500 ms erfolgt eine Überprüfung ob
  - alle Leichtgewichtsprozesse in dem Zyklus ausgeführt wurden
  - die Endzeit eines vorhergehenden Leichtgewichtsprozess vor der Startzeit eines nachfolgenden Leichtgewichts-prozesses liegt
- Bei Verletzung der Zeiteigen-schaften werden geeignete Maß-nahmen ergriffen
- Zeitfunktionalitäten und Schnitt-stelle werden durch Basisklasse zur Verfügung gestellt
- Der Überwachungsprozess ist durch einen Watchdog abgesichert

State-Diagramm Sternenkamera-Protokoll



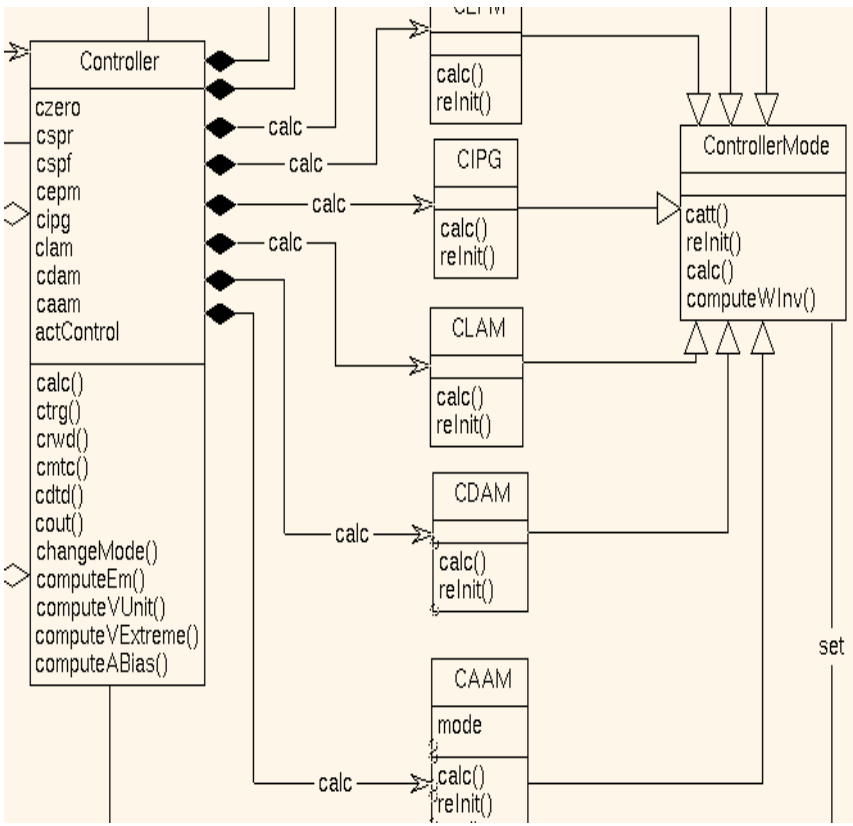
- State-Diagramme sind sehr gut zur Visualisierung von Zustandsabläufen geeignet
- State-Diagramme werden auch von Nicht-Informatikern verstanden
- Eine Modellierung die eine automatische Codegenerierung erlaubt, wird bei komplexeren Systemen unübersichtlich

Implementierung einer Zustandsmaschine



- Zustandsmaschinen lassen sich einfach auf Objekte abbilden
- Die Zustandsmaschine besitzt Instanzen der Zustände
- Die Zustände besitzen Instanzen der ausgehenden Transitionen
- Lässt sich leicht um weitere Zustände erweitern

Kapselung von Regelungsstrategien



- Starke Trennung der Regelstrategien ermöglicht
  - Test lassen sich einfach auf eine Regelstrategie beschränken ohne das andere Regelstrategien bereits implementiert sind
  - geringere Verschachtelungstiefe der Software
  - leicht erweiterbar um weitere Regelstrategien ohne zusätzliche Fehlerquellen in vorhandenen Code zu verursachen
- Berechnungen welche mehrere Regelungsstrategien verwenden lassen sich in der Basisklasse bereit-stellen

Schwächen der UML

- Einbinden der UML-Diagramme in die nach ECSS geforderte Dokumentation
- Kein Stereotyp zur Verknüpfung der UML mit den Requirements und Generierung einer Traceability-Matrix
- Komplexere Zusammenhänge werden leicht unübersichtlich
- Es fehlen unterschiedliche Abstraktionsebenen und/oder –sichten
- Teilweise waren Stereotypen in Diagrammen nicht sichtbar

Stärken des objektorientierten Ansatzes

- Starke Kapselung schützt Systembestandteile vor unzulässigen Zugriffen
- Schnittstellen lassen sich durch Klassen zur Verfügung stellen
- Klare Struktur vereinfacht die Software-Analyse
- Wiederverwendung von Klassen vorhandener Systembestandteilen
- Kapselung von Kommando-Handlern und Regelungsstrategien ermöglicht Erweiterung des Systems ohne Fehler in anderen schon getesteten Systembestandteilen zu verursachen