Dependable Systems made by FIRST
BOSS: Real time Operating System in Space

Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

Sergio Montenegro
FhG FIRST
www.first.fhg.de/~sergio
sergio@first.fhg.de
BIRD Aufnahmen
BIRD-Satellit (BIRD : Bi-spectral InfraRed Detection) DLR

A Nutzlastsegment
B Elektroniksegment
C Dienstsegment
1 Sonnensensor
2 Infraotsystemrodictor
3 GPS-Antenne
4 Sternsensor-1
5 S-Band-Halbunndstrahltante
6 Zweikanal-Infrarotsensoreystem
7 Energiekontrollseinheit
8 Bordcomputer (Satellitenlekttronik)
9 Reaktionsroder (M)
10 Sattelitenroductor
11 2 x 4 NiH2-Zellen (12 Ah)
12 Solarzellenaufklepsübender
13 S-Band-Einstrahlantenne
14 S-Band-Elektronik (Sendet)
15 Wärmerohr
16 Matrikkamera
Real time embedded operating system

Design for dependability

Design for formal verification

Support for fault tolerance

Fast, small,

.... and .... Open Source!
BOSS... designed for dependability

1. Irreducible complexity
2. Framework technology to reduce complexity
3. Component technology to handle complexity (not to create complexity)

-> + Formal verification
Complexity destroys safety

- Required mental capacity
- Safety limit
- Forbidden area
  - To complex to be safe
- Human mental capacity
- Design and implementation errors have their roots in the high complexity
- Complexity of the underlying System
- Komplexity

Eg. BIRD: To complex to exist?

Fraunhofer Institut Rechnerarchitektur und Softwaretechnik
Simple -> Formal Verification

BOSS basic functions (for every thing): lists
   Operations:
   Insert in list
   remove from list
-> So easy, so safe
-> application of formal methods becomes possible
    Modell checker
    Theorem prover
Simple!
10kB Foot Print
BOSS... designed for dependability

1. Irreducible complexity
2. Framework technology to reduce complexity
3. Component technology to handle complexity (not to create complexity)

-> + Formal verification
OS Framework:

modern software technology / engineering
Design for real time safety critical applications
cost effective
BOSS... designed for dependability

1. Irreducible complexity
2. Framework technology to reduce complexity
3. Component technology to handle complexity (not to create complexity)

-> + Formal verification
Complexity mastering by using Components

Build the System by plugging applications as components

Communication by using Software buses and routers
Middle Ware (1): What you program
Middle Ware (2): What you can get

- Distributed applications
- Dynamic reconfiguration
- Redundancy management
Middle Ware (3): What you can get

Fault tolerance support
multiple voters
monitors
TMR and beyond
Distributed FT
Fault tolerance support
multiple voters
monitors
TMR and beyond
Distributed FT
BOSS Layers and Portability
BOSS... designed for dependability

1. Irreducible complexity
2. Framework technology to reduce complexity
3. Component technology to handle complexity (not to create complexity)

-> + Formal verification
BOSS ... is it correct?

Operational "Specification" ... How to do ...

BOSS + Applic. C++

HOL semantics of C++

Gcc Front-End
Code Generator

Machine Code

Execution/Test

Compiler Checker

BOSS ...

HOL

Operat. Spec.

HOL Generator

Valid: Yes/No

Consistent: Yes/No

Satisfied: Yes/No

Logfile with Time

Isabelle Checker

Timing Checker

Axiomatic Specification: What to do What not to do

Universität Karlsruhe

Fraunhofer Institut Rechnerarchitektur und Softwaretechnik

TU Berlin
Thank You