



# Program Analysis

Dr. Florian Martin  
AbsInt

[AbsInt.com](http://AbsInt.com)

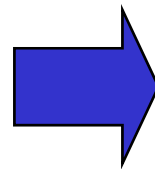
# Introduction

- For Questions:
  - ask immediately
  - mail [florian.martin@absint.com](mailto:florian.martin@absint.com)
  - mail [esd07-tutor@gigasun.cs.uni-sb.de](mailto:esd07-tutor@gigasun.cs.uni-sb.de)

# Program Analysis (PA)

- Classical compiler technique
- Prove properties about programs to generate more efficient code
- Example: **Loop Invariant Code Motion**

```
for (i=0;i<10;i++) {  
    z = r*r;  
    k = k*i;  
}
```



```
z = r*r;  
for (i=0;i<10;i++) {  
    k = k*i;  
}
```

# PA in Embedded Systems

PA to check the **non functional correctness** of the program

- Pointer errors
- Overflow
- Division by zero
- Precision analyzer
- Stack overflow
- Timing analysis

# Ariane 5



# Ariane 5

- June 4, 1996
- Rocket with cargo valued **\$500 million**
- error in the inertial reference system
- 64 bit floating point number relating to the horizontal velocity was converted to a 16 bit signed integer
- The number was larger than 32,767, the largest integer storable in a 16 bit signed integer, and thus the **conversion failed**.

# PolySpace Verifier

Now [www.mathworks.com/products/polyspace/](http://www.mathworks.com/products/polyspace/)

Analyzer for C programs.

## Typical Run-Time Errors Detected

- [Overflows and underflows](#)
- Division by zero and other arithmetic errors
- Out-of-bounds array access
- Illegally dereferenced pointers
- Read-only access to noninitialized data
- Dangerous type conversions
- Dead code
- Access to null this pointer (C++)
- Dynamic errors related to object programming and inheritance (C++)
- Errors related to exception handling (C++)
- Noninitialized class members (C++)

# PolySpace Verifier

- **Green:** Proven reliable under all operating conditions
- **Red:** Proven faulty each time the operation is executed
- **Grey:** Proven unreachable (may indicate a functional issue)
- **Orange:** Unproven code section (a run-time error might occur under certain operating conditions)



# Patriot

- 25/02/91: a Patriot missile misses a Scud in Dharan and crashes on an american building : 28 deads.
- Cause :
  - the missile program had been running for 100 hours, incrementing an integer every 0.1 second
  - but 0.1 not representable in a finite number of digits in base 2
  - $1/_{10} = 0.00011001100110011001100 \dots$

Truncation error ~ 0.000000095 (decimal)

Drift, on 100 hours ~ 0.34s

Location error on the scud ~ 500m

# Fluctuat

Static analyzer for C Programs with floating-point computations

Can compute

- Bounds of floating-point values
- Bounds on the discrepancy error between the real and floating-point computations
- If possible, the main source of this error

# Altona Signal Box (=Stellwerk)

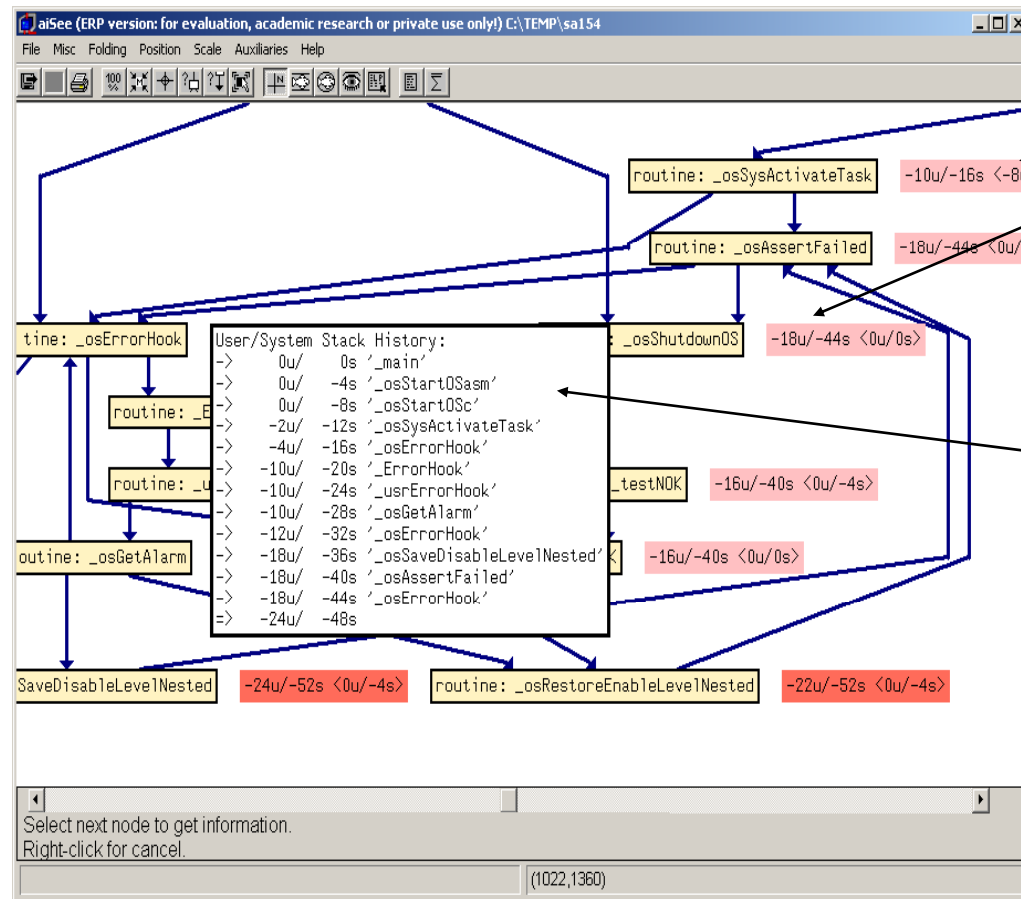
1995 the old signal box in Hamburg Altona was replaced by a 486 real time system. The machine crashed several times and a reboot took 10 min.

**The whole station was closed.** This had an impact on many trains in Germany.

After two days the bug was found: a **stack overflow**



# StackAnalyzer



# StackAnalyzer

StackAnalyzer Control Center

Messages Results Options About

Task	Stack Usage	Sum of	Max of
basicTaskFirst_system	36	Σ	
CallGraph [_basicTaskFirstfunc] [System]		8	
max StackOffset1		22	max
FastTimerIsr_system			22
StaticOffset		6	
basicTaskFirst_user	36	Σ	
CallGraph [_basicTaskFirstfunc] [User]		4	
max StackOffset1		0	max
FastTimerIsr_user			0
StaticOffset		32	
basicTaskSecond_system	16	Σ	
basicTaskSecond_user	34	Σ	
basicTaskThird_system	8	Σ	
basicTaskThird_user	0	Σ	
extendedTaskFirst_system	8	Σ	
extendedTaskFirst_user	6	Σ	

Current Project: gen2.afg

Analyze Display Close

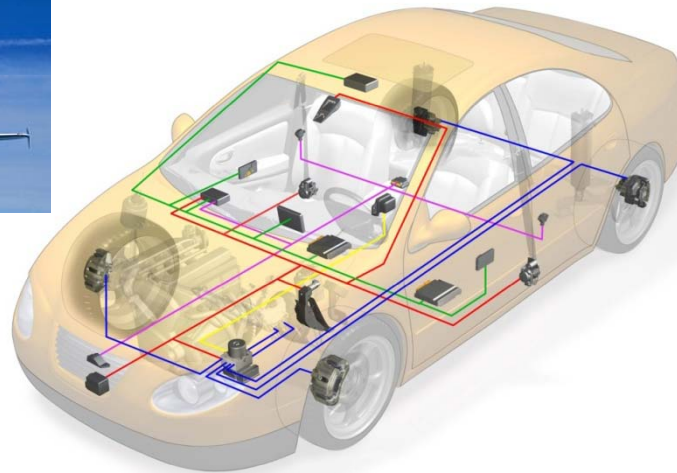
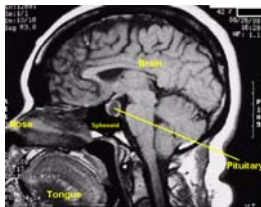
(C16x/ST10)

Annotations:

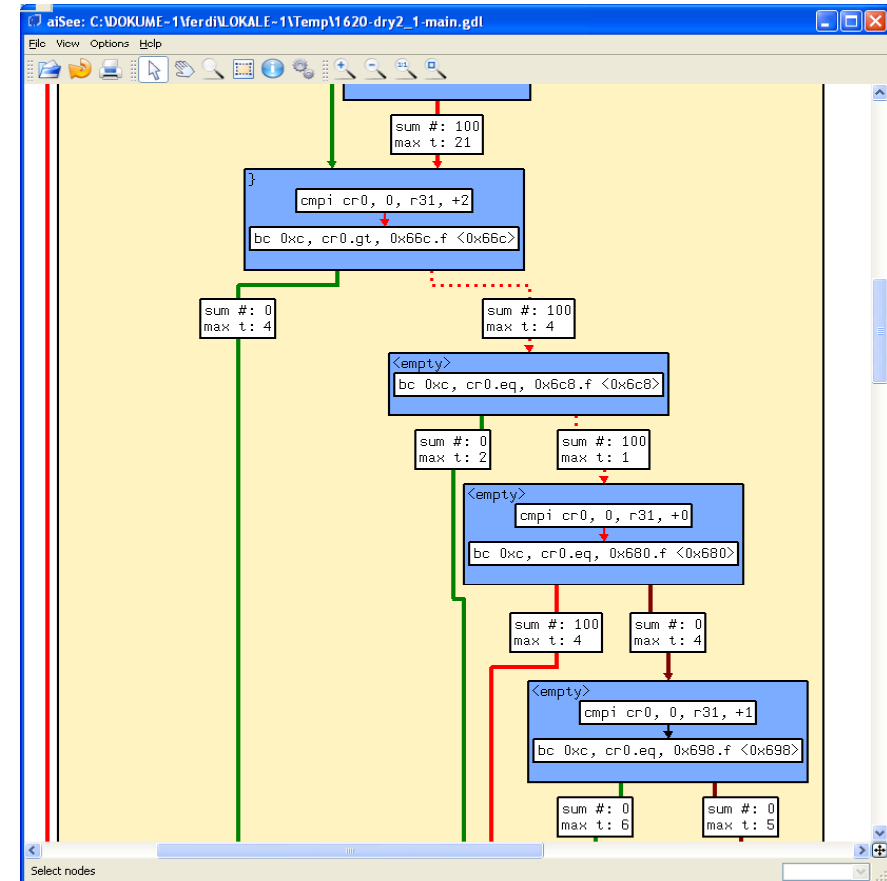
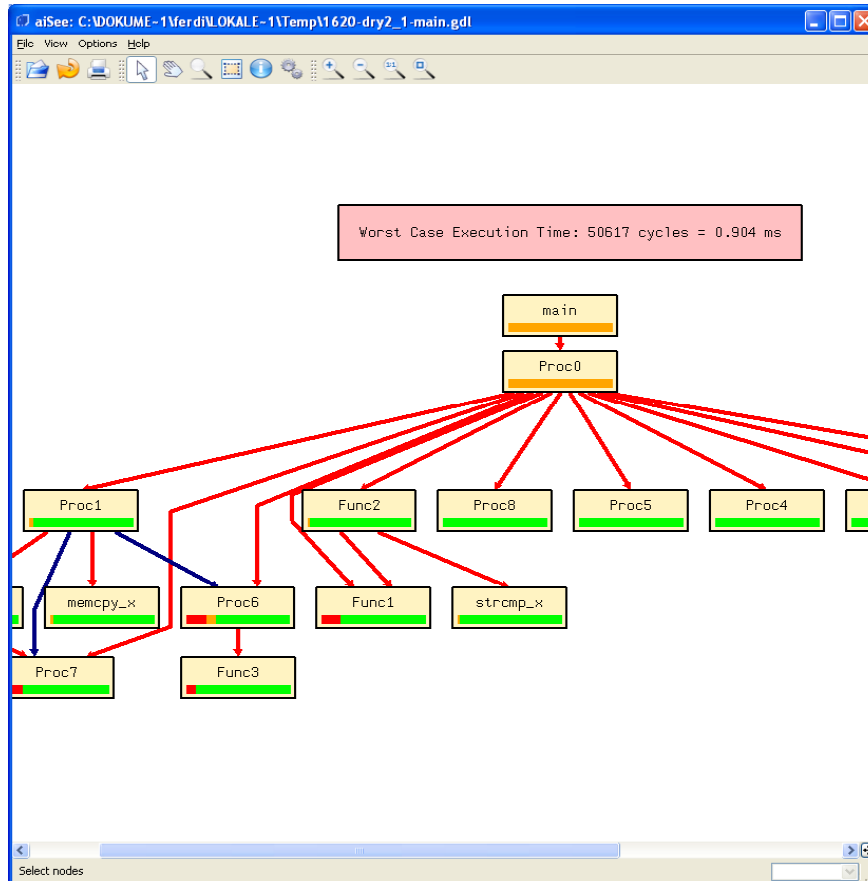
- Maximum system stack usage
- Category 1 ISR
- Task context offset
- Maximum User stack usage

# Hard Real-Time Systems

- Controllers in planes, cars, plants, ... are expected to finish their tasks within reliable time bounds.
- Schedulability analysis must be performed
- Hence, it is essential that an upper bound on the execution times of all tasks is known
- Commonly called the Worst-Case Execution Time (WCET)

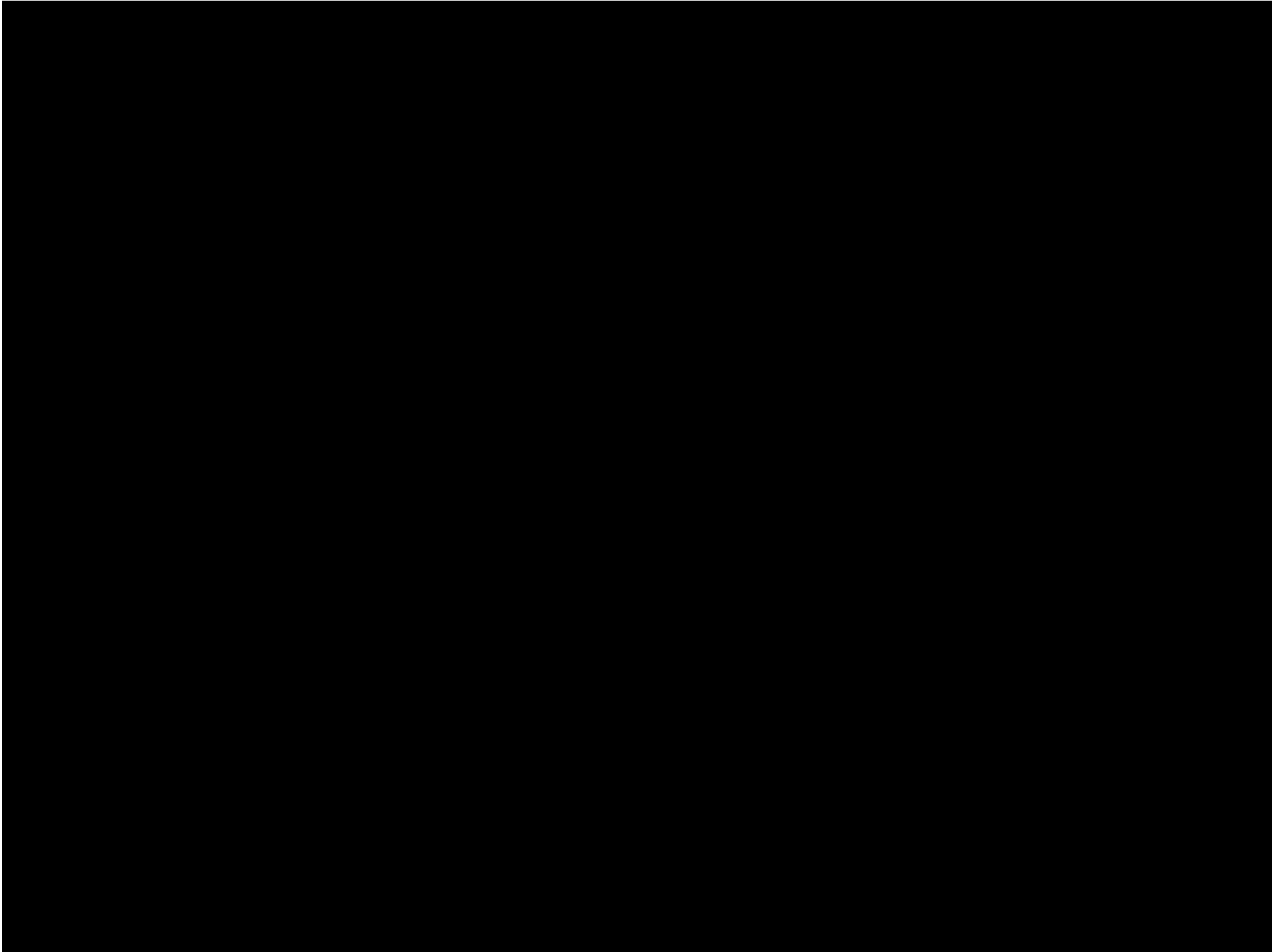


# aiT: Timing Details

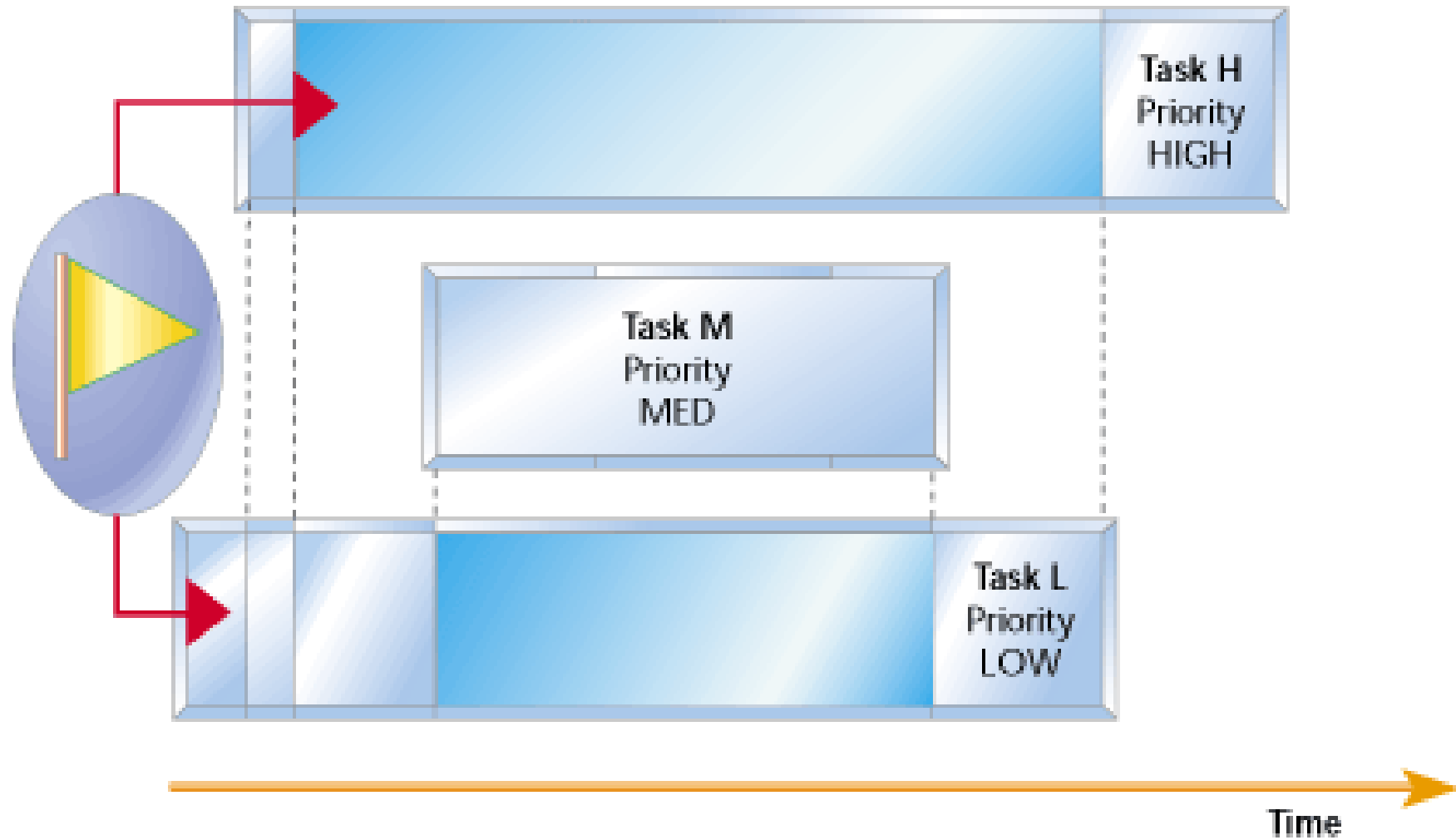


# Analysis reports

Results				
Control file				
Name	ID	Entry	Result	Comment
[-] Main	7		2914	add
[-] MainNoPrime	4	main	201	
[-] PrimeT	6		2713	max
[-] Operation: add			893	
[-] PrimeN	1	prime	2713	With all subroutines
[-] MainNoPrime	4	main	201	
[-] PrimeN	1	prime	2713	With all subroutines
[-] PrimeT	6		2713	max
[-] Operation: add			893	
[-] PrimeW	2	prime	776	Without even and divides
[-] Swapl	5		117	add
[-] PrimeN	1	prime	2713	With all subroutines
[-] PrimeW	2	prime	776	Without even and divides
[-] Swap	3	swap	85	
[-] Swapl	5		117	add
[-] Swap	3	swap	85	
[-] int			32	



# Pathfinder (Priority Inversion)



# Pathfinder (Priority Inversion)

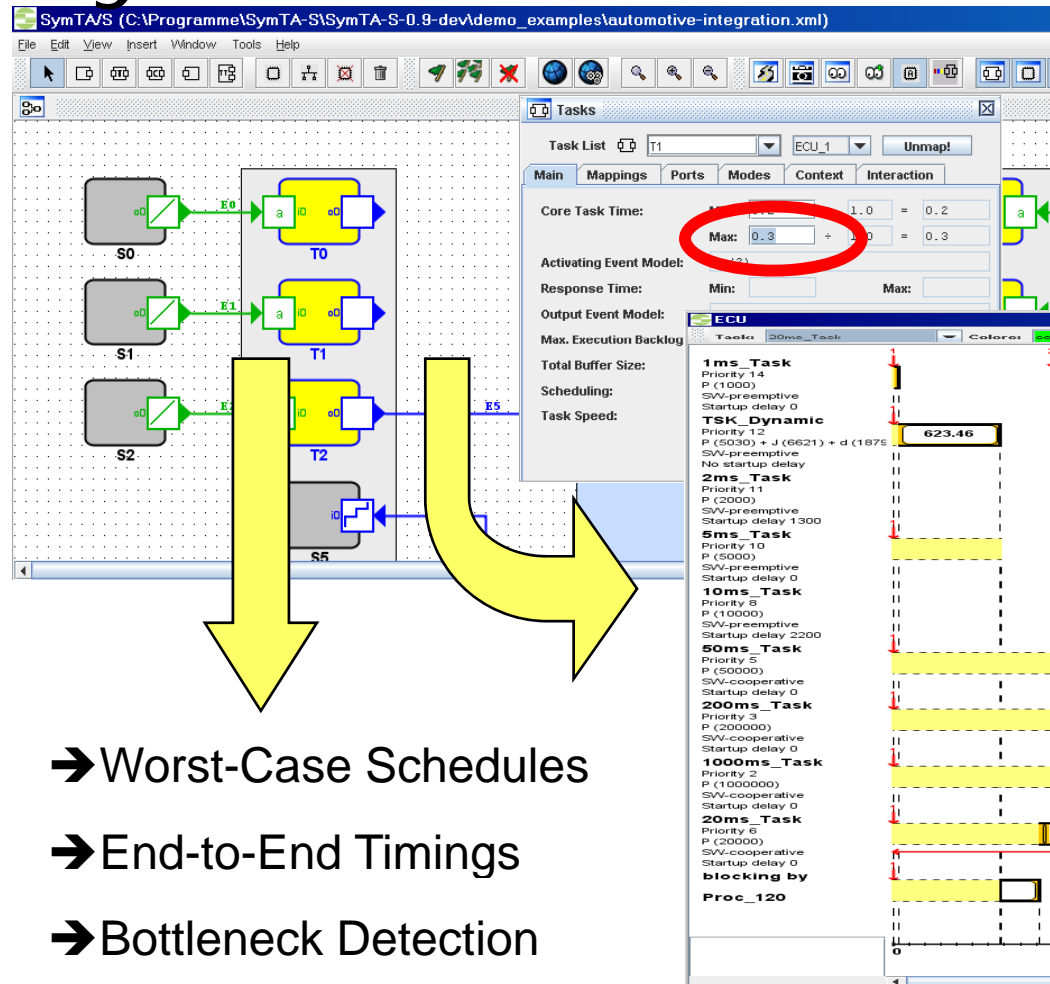
Picture taken from

<http://www.netrino.com/Embedded-Systems/How-To/RTOS-Priority-Inversion>

Further reading:

- [http://en.wikipedia.org/wiki/Priority\\_inversion](http://en.wikipedia.org/wiki/Priority_inversion)

# SymTA/S



SymTA/S by  
 SYMTA VISION

- Worst-Case Schedules
- End-to-End Timings
- Bottleneck Detection
- System Optimization

# Software Failures

- Collection of Software Bugs at <http://www5.in.tum.de/~huckle/bugse.html>

# Additional Reading

- Nielson, Nielson, Hankin:  
Principles of Program Analysis  
Springer
- PAG/WWW  
<http://www.program-analysis.com>