Tuning Development of Distributed Real-Time Systems with SDL and MSC:

Current Experience and Future Issues

"Eighth SDL Forum"

(SDL '97)

Evry, France

September 23 - 26, 1997

Rainer Gerlich
BSSE System and Software Engineering
Auf dem Ruhbuehl 181
D-88090 Immenstaad
Phone: +49/7545/91.12.58
Mobile: +49/171/80.20.659
Fax: +49/7545/91.12.40
e-mail: gerlich@t-online.de

BSSE System and Software Engineering

Foreword

❐ Overloading of terms

- performance and tuning
  1. resource utilisation
  2. efficiency of system development
  3. quality of a system and ***early*** system validation

❐ Message

- SDL is helpful for such system tuning and increasing of performance
- however, use of SDL and SDL tools needs to be tuned

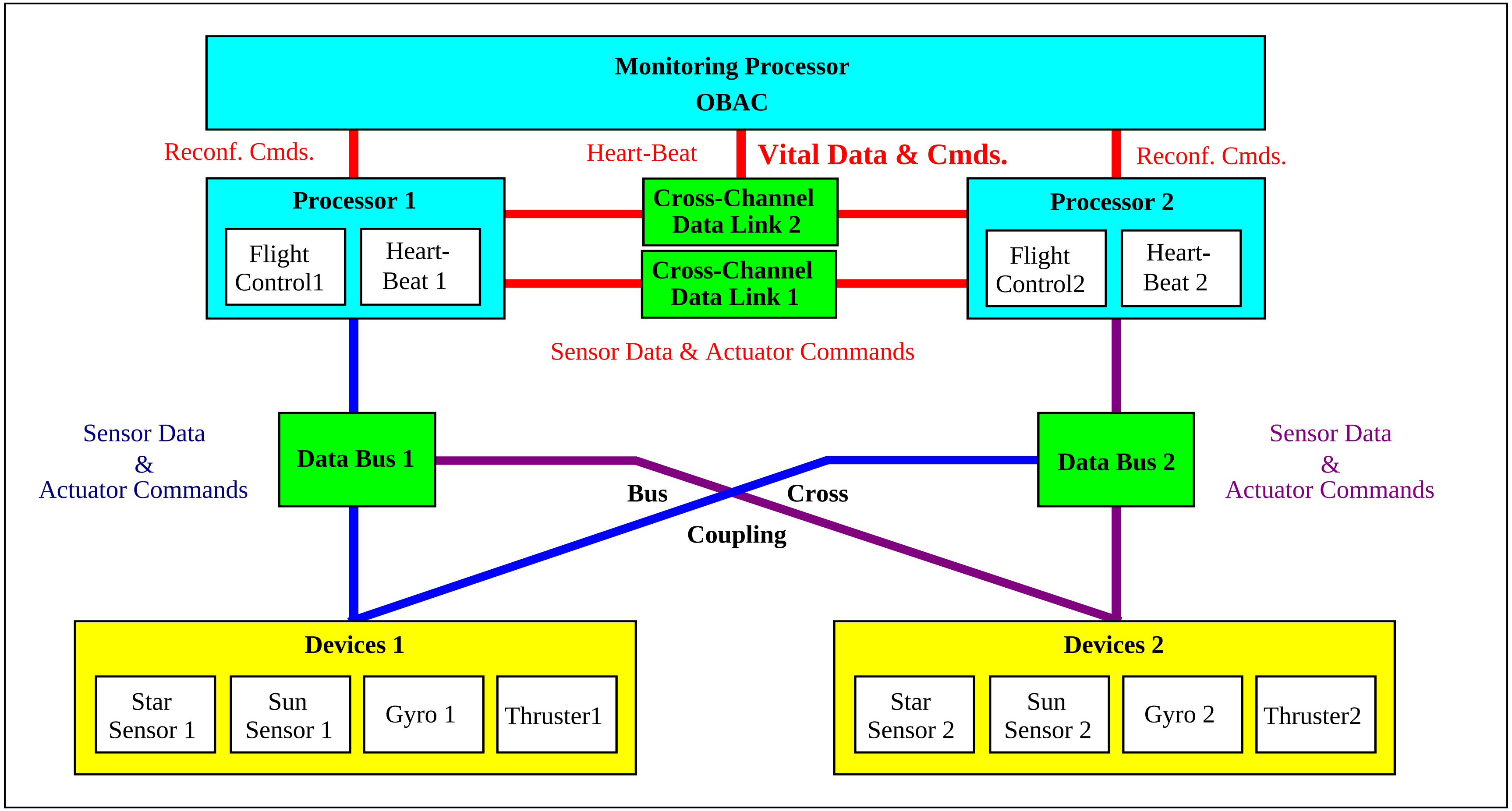
SDL and Heterogeneous Distributed (Fault-Tolerant) Systems (1)

- **which type of application**
  - SDL not necessarily limited to "classical" protocol applications

  e.g.
  - embedded real-time systems (OS like VxWorks, space application)
  - client-server systems (UNIX-based, Solaris/Sparc, Solarisx86, Linux)

- **optimised usage of SDL**
  - behavioural part of an application (system control) in SDL
  - functional part in C / Ada

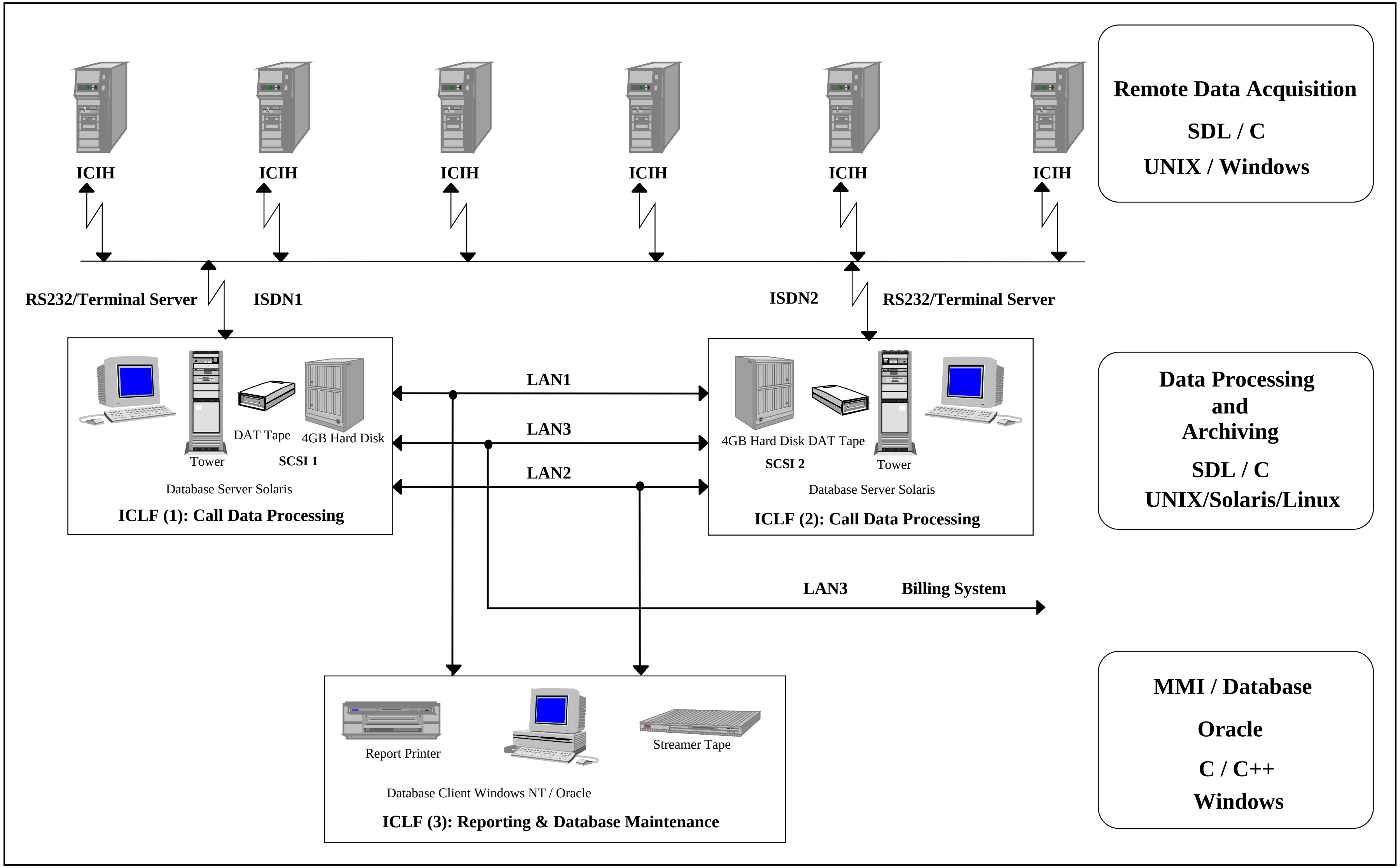
Example: Embedded On-Board Space Real-Time System



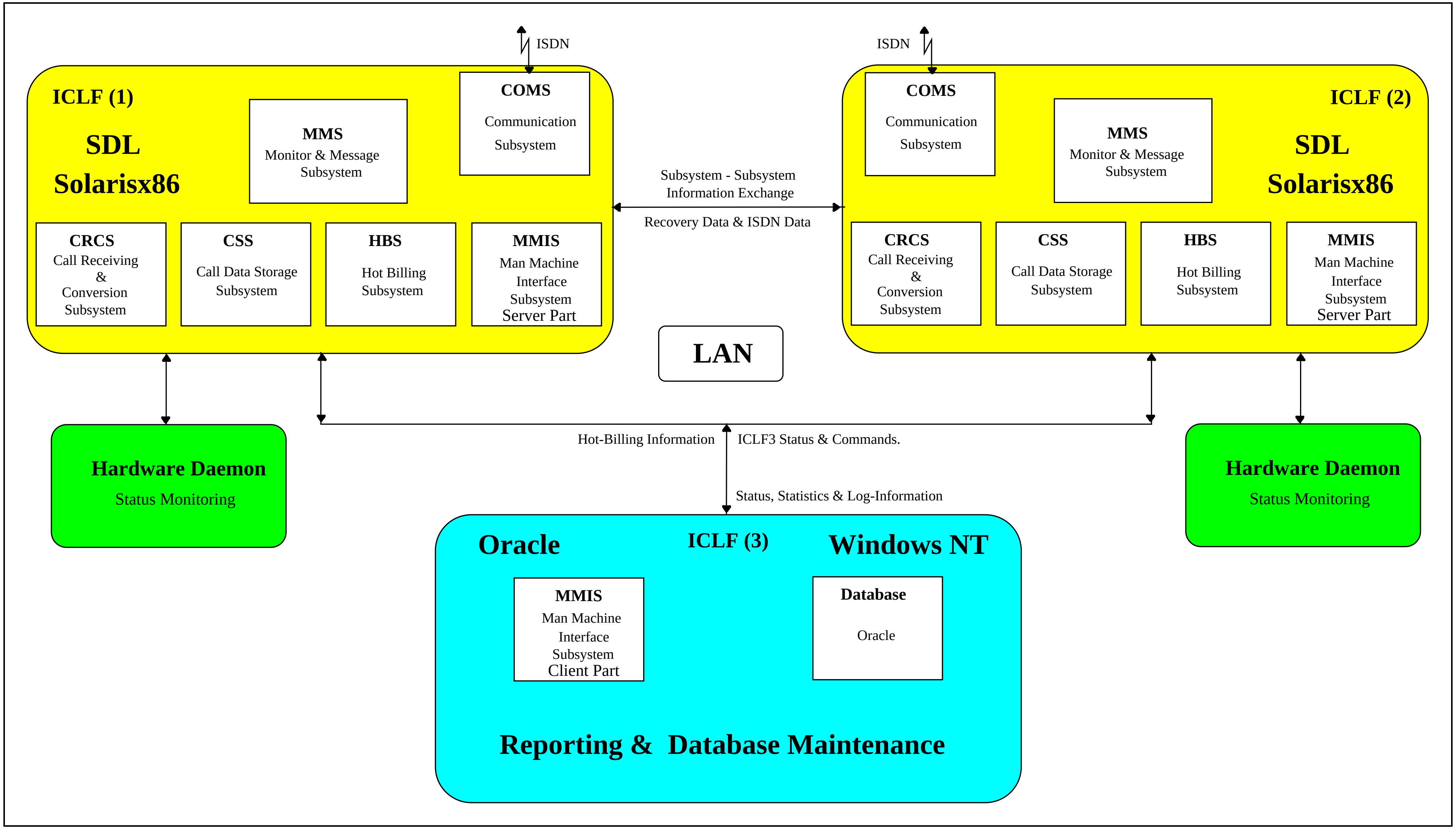
case study: coherent transition from specification to design

from simulation (UNIX) to target system (UNIX, VxWorks)

Heterogeneous Distributed Fault-Tolerant System



Software Architecture



SDL and Heterogeneous Distributed (Fault-Tolerant) Systems (2)

❐ why SDL?

- strong by formalisation of behaviour (FSM)
- strong by formal description of scenarios (MSC)
- strong by verification means (tool simulators, exhaustive simulation)
- efficient by capability of (automated) code generation

❐ customisation needed for more general class of applications

- coherent transition from specification down to target code generation and final system
- **(early) system validation**

Experience

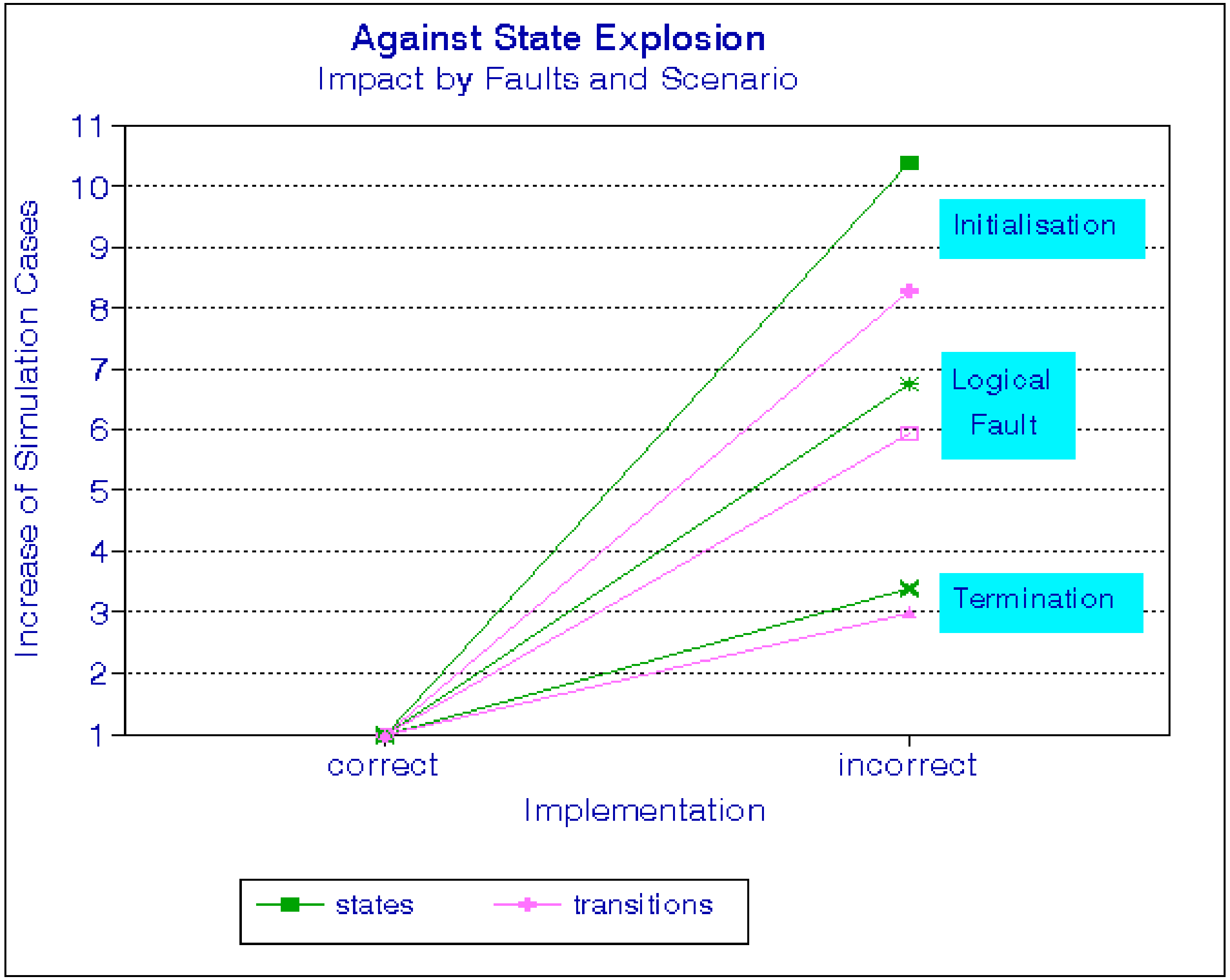
- **space domain**
  - selected for ESA/ESTEC for a new lifecycle approach
    Early System Validation: EaSyVaDe
  - top-down refinement using executable SDL models
  - safety-critical / fault-tolerant real-time embedded systems
  - ***early*** *and* ***complete*** system validation
  - need for behavioural, functional and performance validation
  - extension of SDL and ObjectGEODE towards performance validation and complete system validation

- **enhanced SDL tool environment**
  - experience by ESA/ESTEC studies: EaSySim
  - enhancement and complete new approach: EaSySim II (BSSE)

Lessons learned

- performance needs to be considered already during exhaustive simulation
- specification modelling may no fit with needs of target system
- rules needed for coherent transition down to target code generation
- state explosion prevents to take real benefit of SDL strongness
- specific modelling approach needs to be supported to reduce number of system states as supported by EaSySim II
- introduction of performance (timing) reduces number of system states

Impact by Logical Faults and Performance Aspects

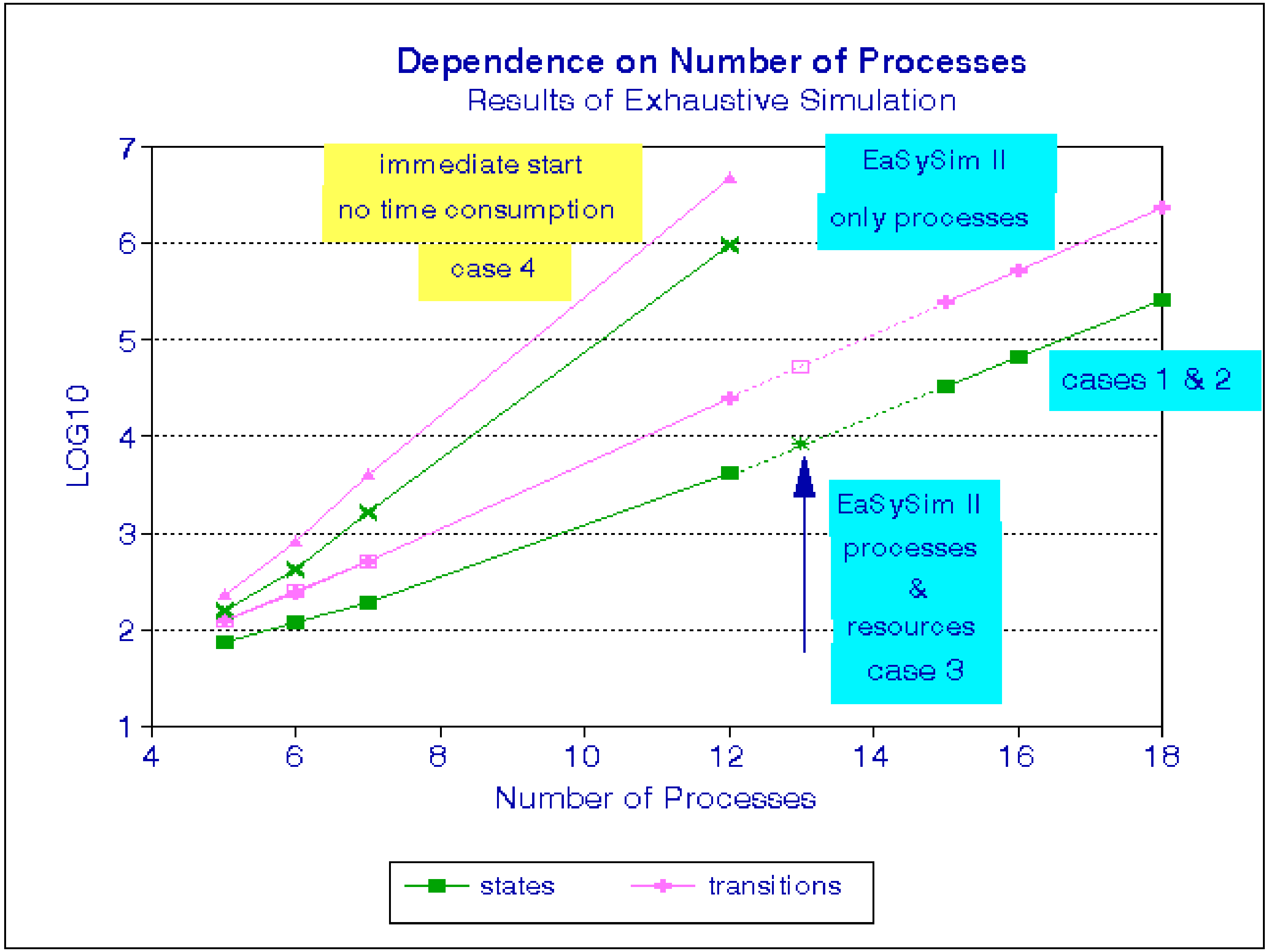


without performance

+ SDL subset

performance impact

Dependence of System States on Number of SDL Processes



EaSySim II

reentrant approach:

helps to escapes

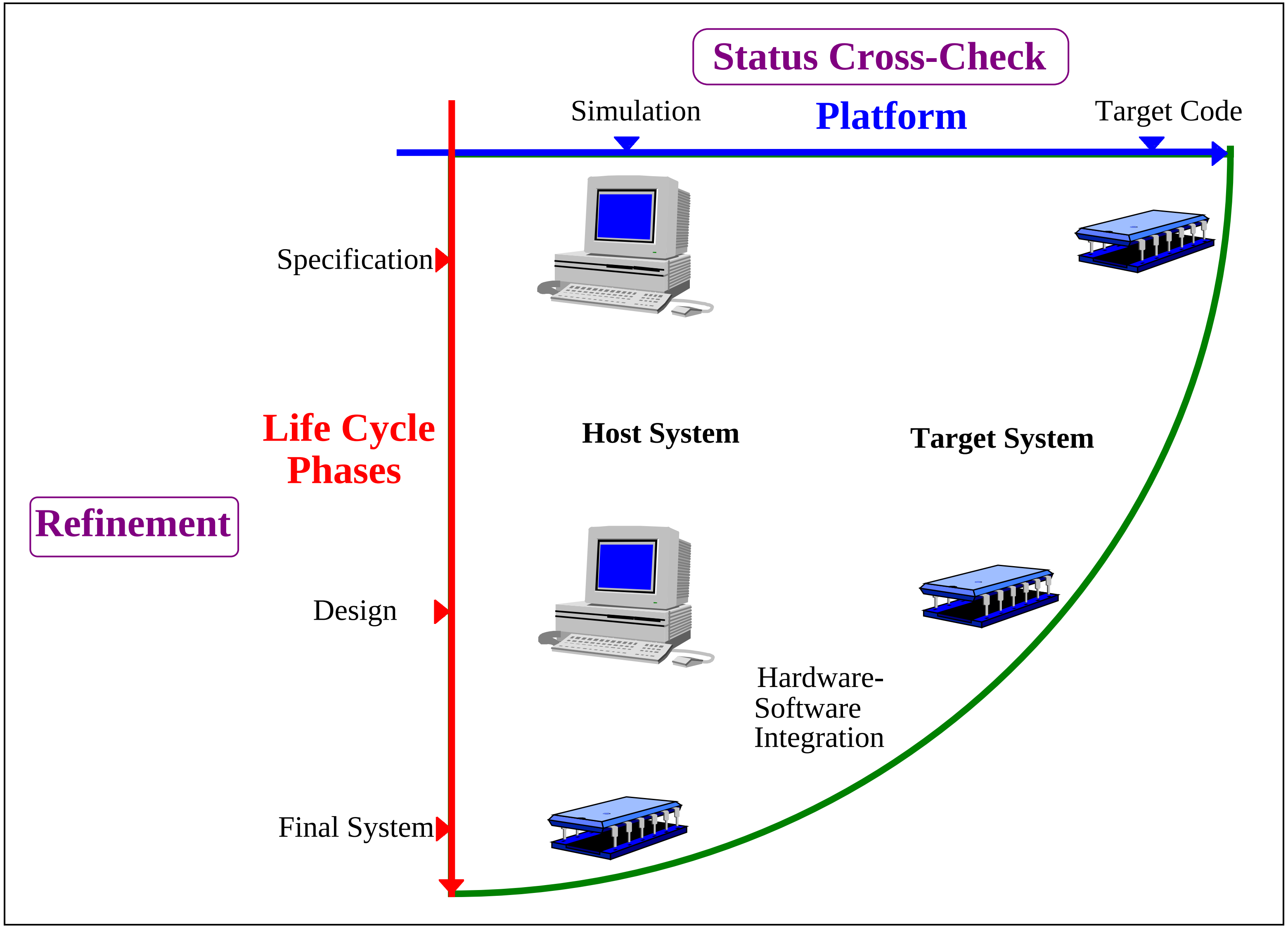
from this limitation

Problems with Correct System Validation

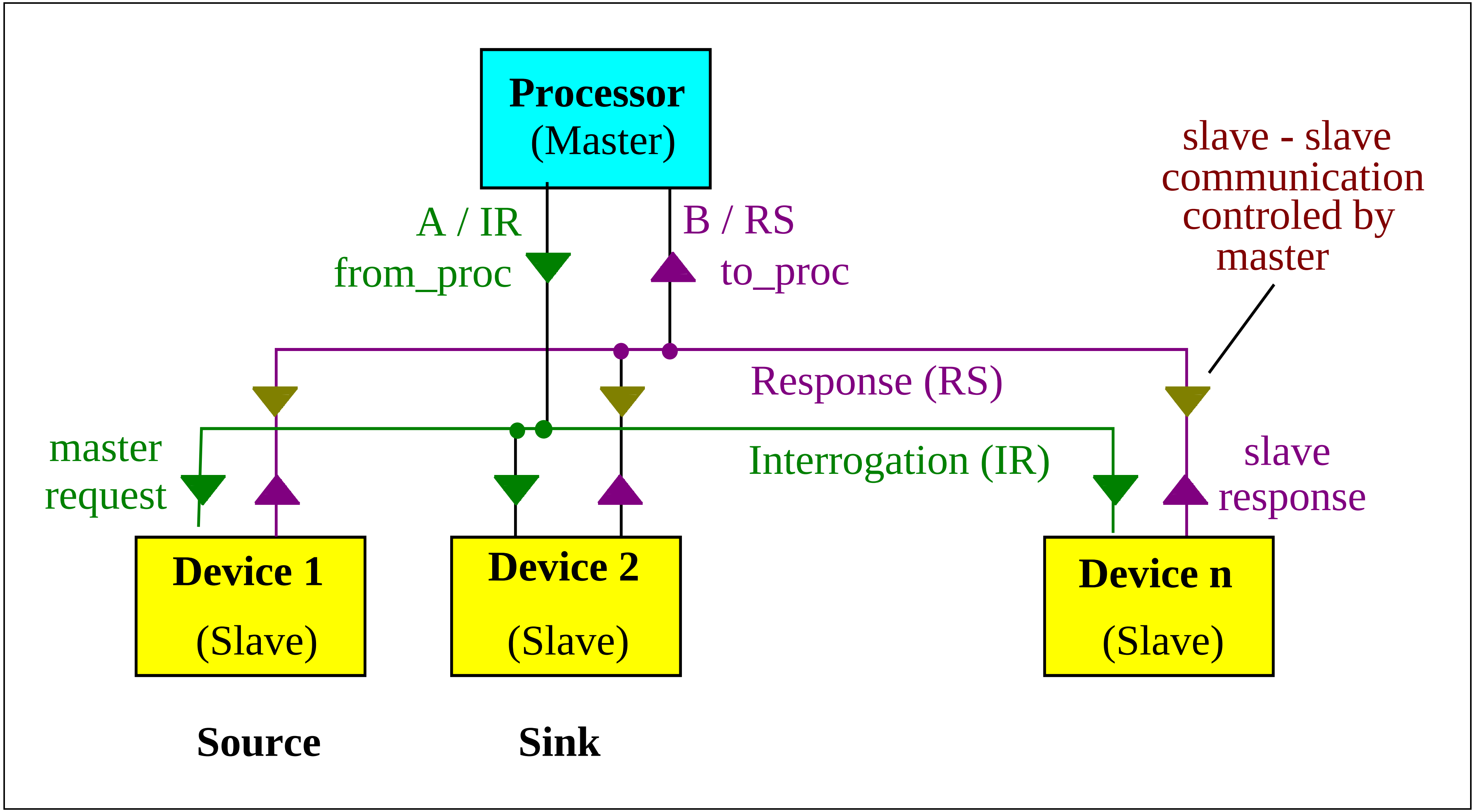
- **Representative modelling**
  - verification / validation methods / tools only respond to a given model
  - performance aspects (timing, resource usage) significantly impact results
  - validation of a high level model does not mean anything
  - response is needed from the system in the real environment
  - in the real environment a system tells you if you are right or wrong

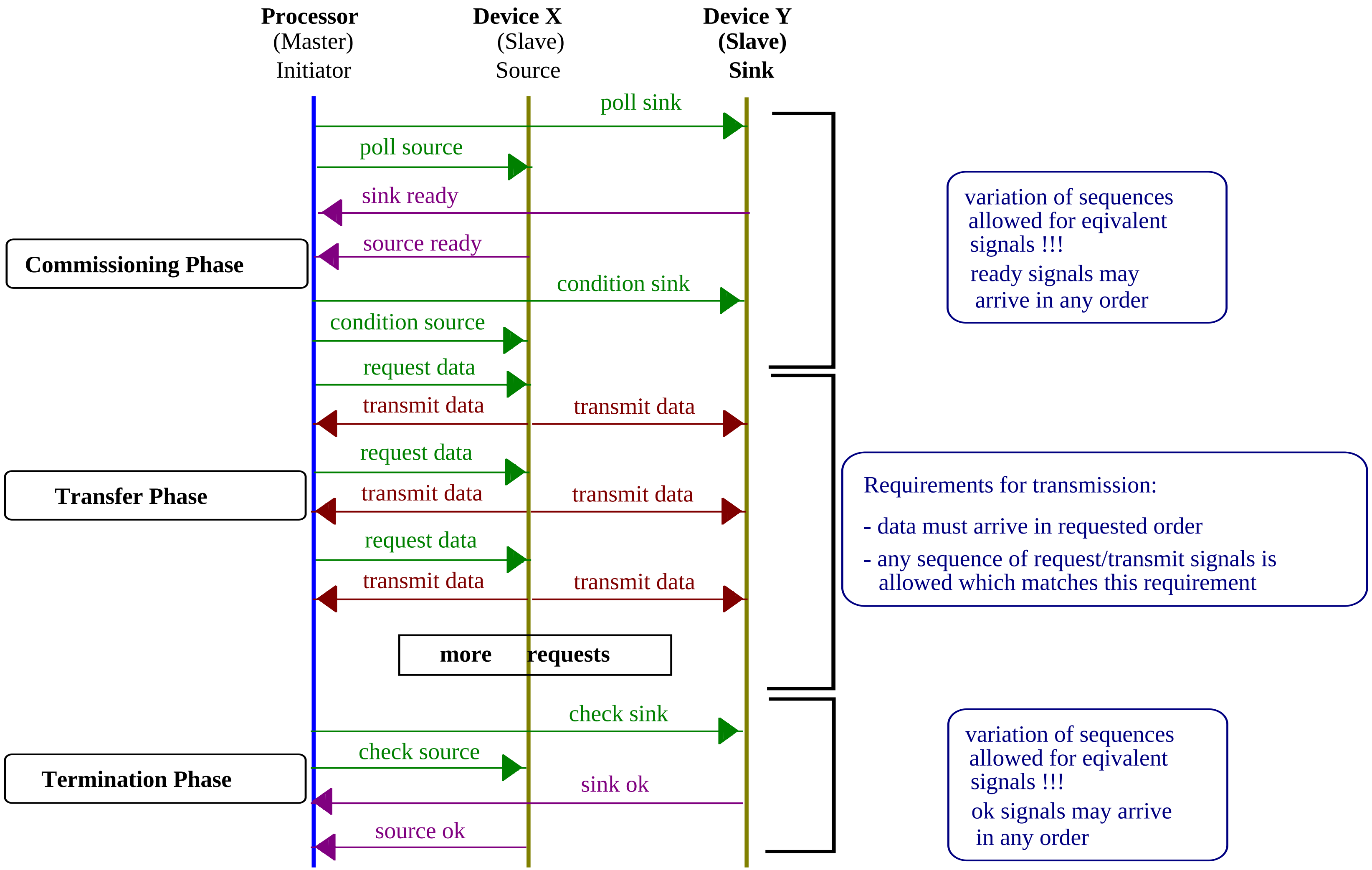
- **if not**
  - it is possible to succeed for an erroneous system, still believing it is correct

The 2-Dimensional Life Cycle

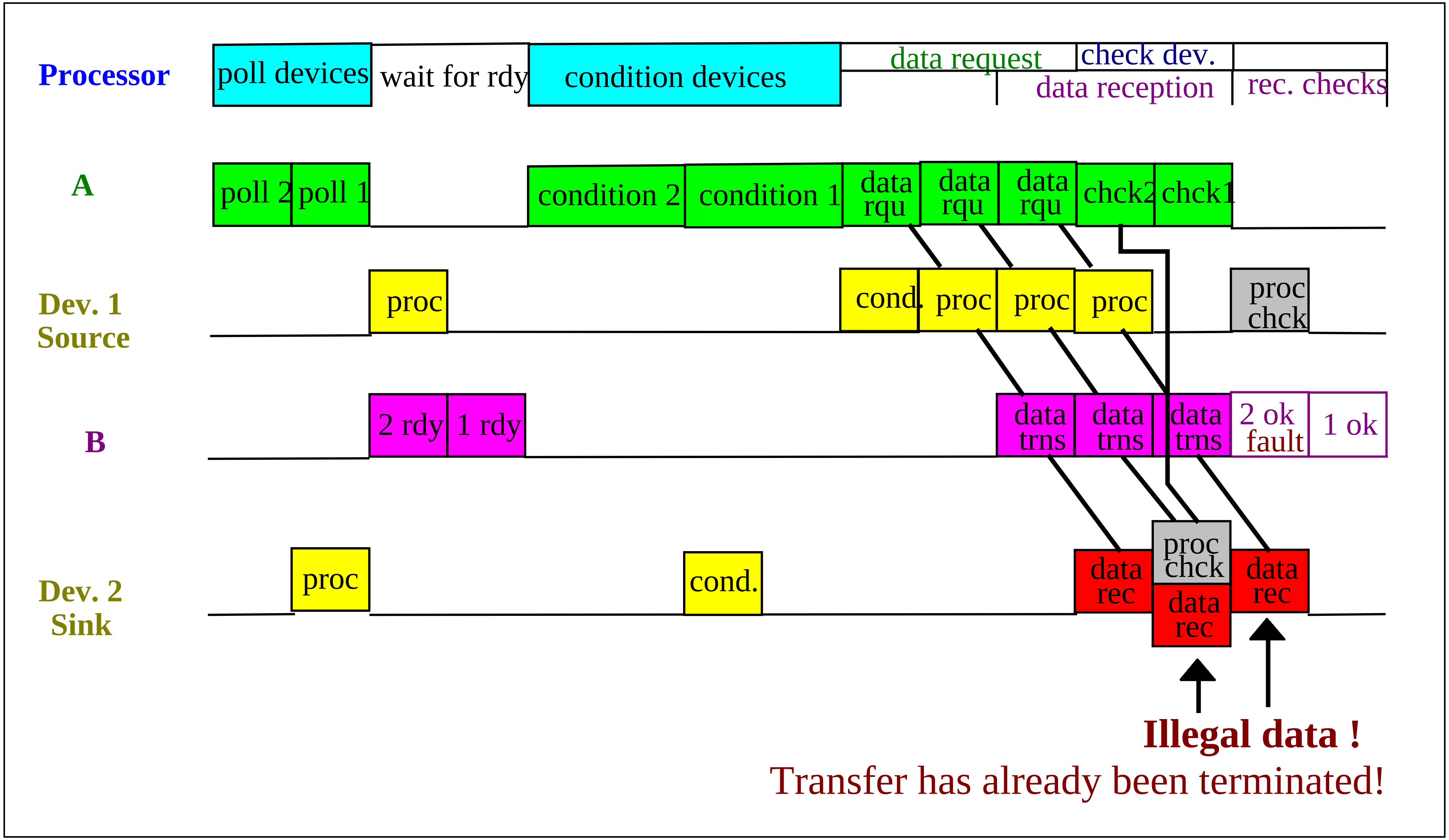


Validation Example System Architecture

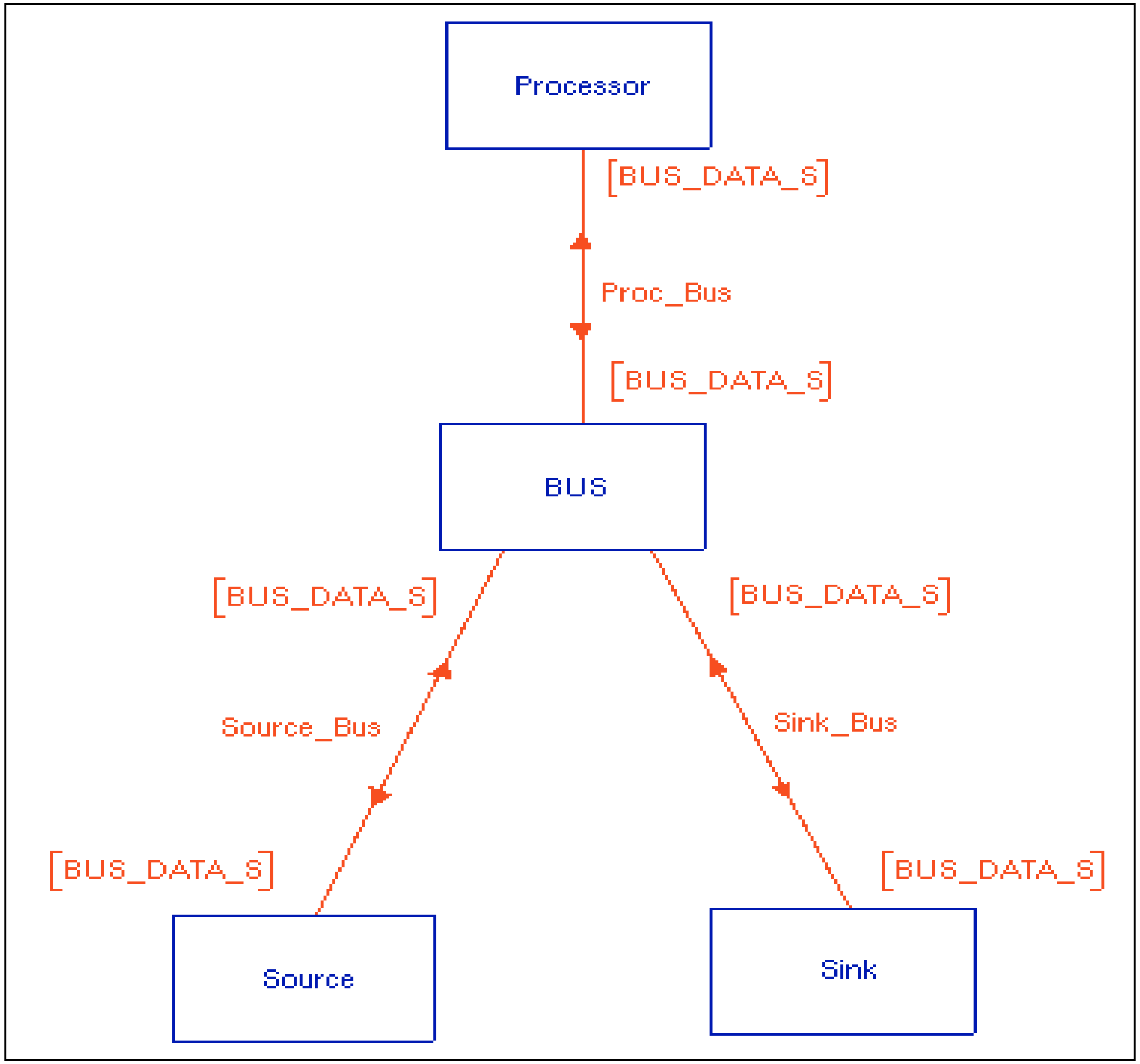




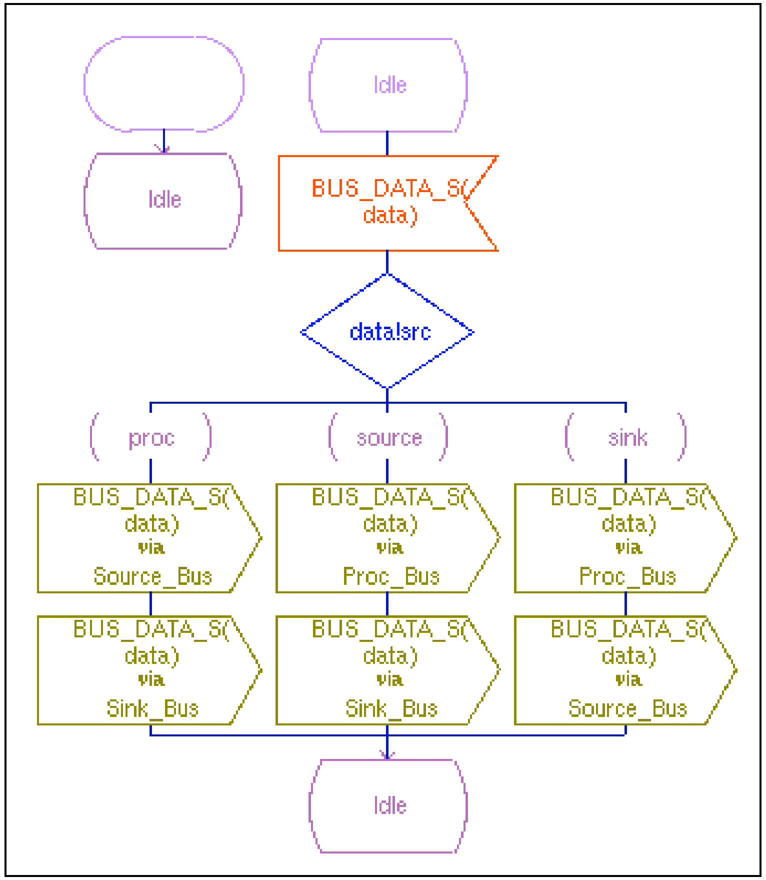
Problem:
Due to varying propagation times in a network the inital sequence of signals may not be the final sequence!


Processor
poll devices
wait for rdy
condition devices
data request
check dev.
data reception
rec. checks
A
poll 2
poll 1
condition 2
condition 1
data rqu
data rqu
data rqu
chck2
chck1
Dev. 1 Source
proc
cond.
proc
proc
proc
proc chck
B
2 rdy
1 rdy
data trns
data trns
data trns
2 ok fault
1 ok
Dev. 2 Sink
proc
cond.
data rec
proc chck
data rec
data rec
Illegal data !
Transfer has already been terminated!

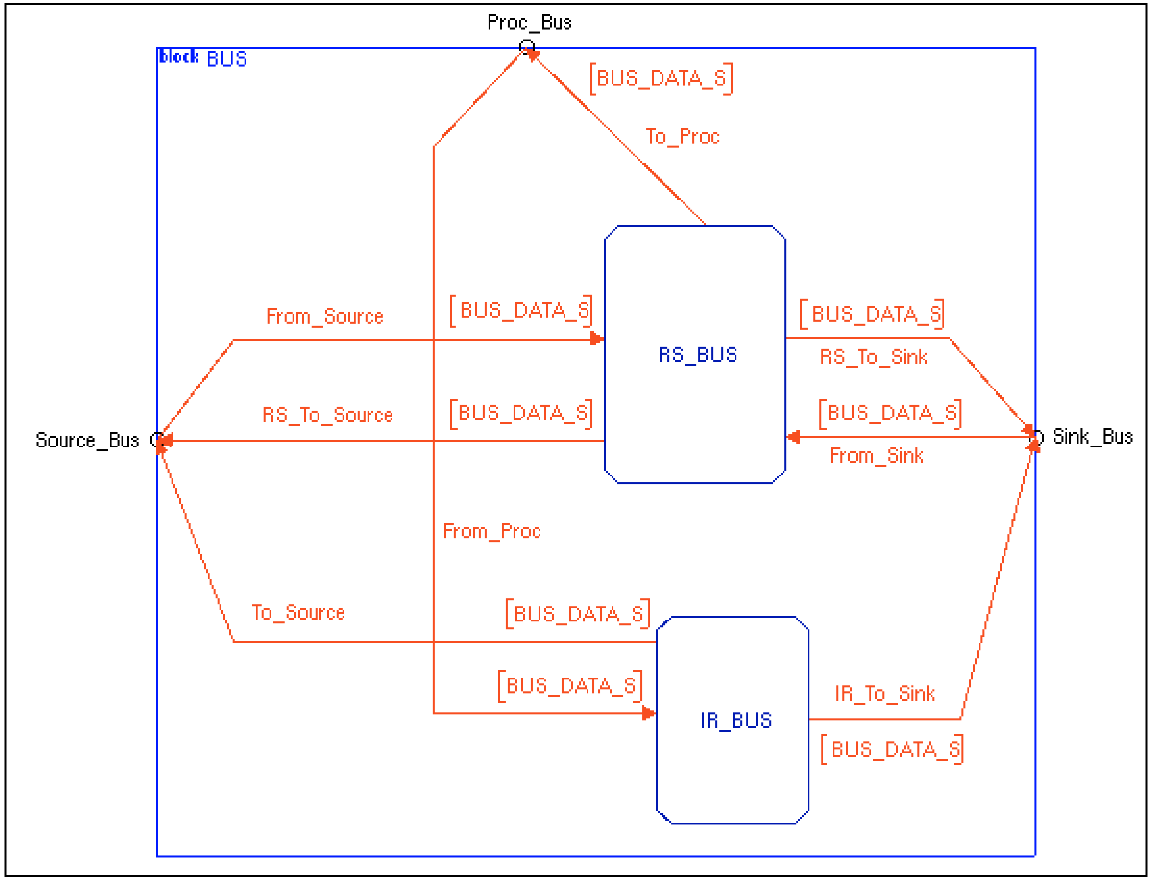
Principal System Structure



Modelling Approach (1)

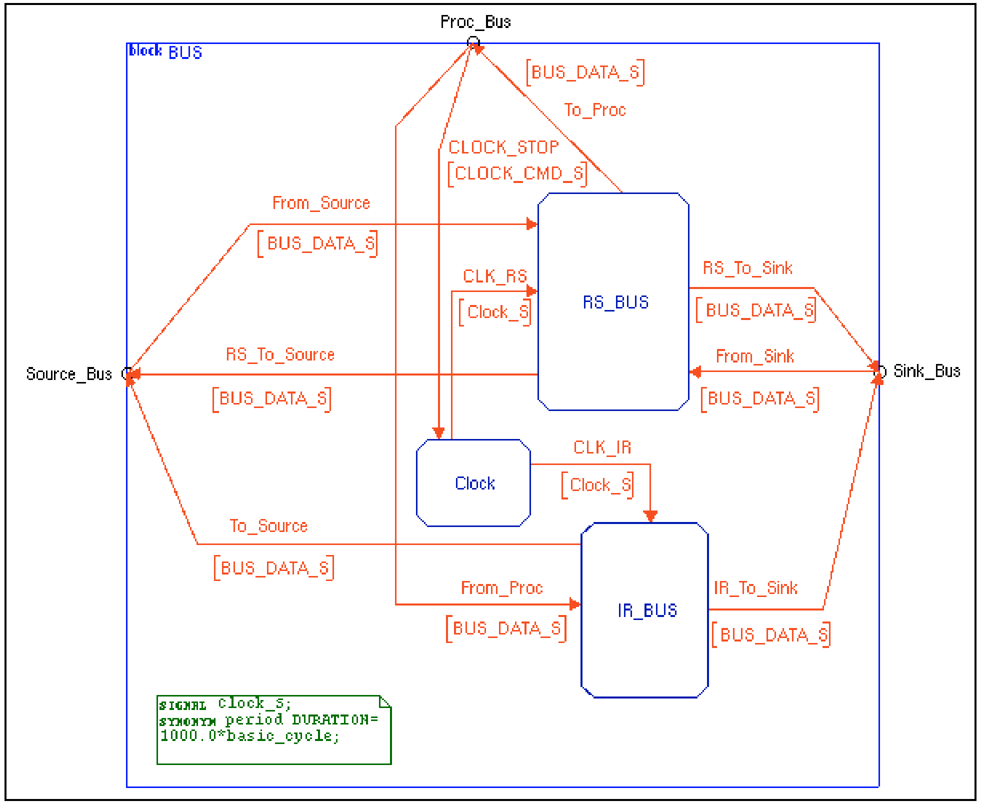


Simple Functional Bi-Directional Bus with Broadcasting

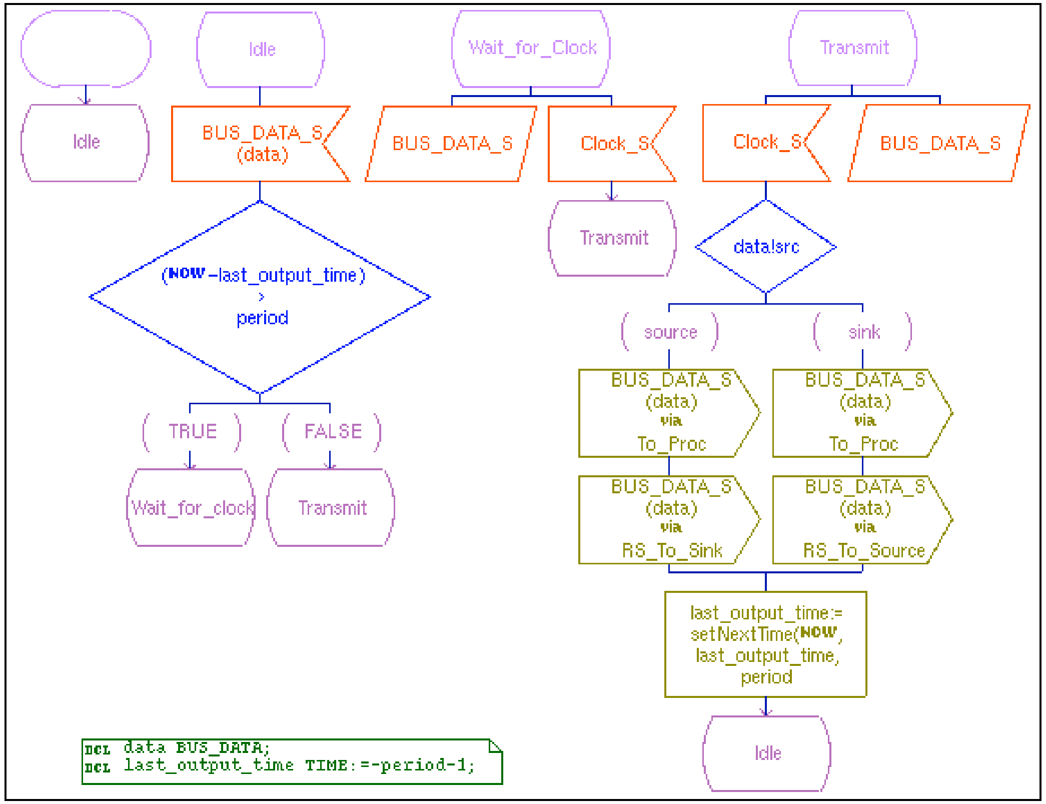


Functional Bus with Two Uni-Directional Bus Lines

Modelling Approach (2)



Fully Representative Architectural Bus



FSM of Uni-Directional Bus Line with Clock (Device-to-Processor Line)

Results (1)

- **question: is the set of SDL states** *(**not** considering performance)* **a superset of system states** *(when considering performance)***?**
  - answer: it is **not**
  - as verified by the "simple" example
  - consequence: results of simulation may give the **wrong** answer

- **modelling task**
  - 9 models at different degree of refinement and representativity

    3 functional and 6 architectural models

    architectural model = refinement towards real system architecture
consideration of timing

Results (2)

❒ evaluation

- models yielding low performance on the target system lead to correct results, 4 out of 9, amongst which the 3 functional models
  low performance = 33% of optimum performance
- verification of 3 out of 6 of the architectural models did not give the correct result
  (predicted "no problems", although there are problems)
- for one architectural model it was correctly predicted that it will not work in the target environment
- for the architectural model yielding optimum performance the result was correct
- when running this correct and optimised architectural model in a functional representation (zero timing) it was rejected as **<u>erroneous</u>**

Results (3)

❒ evaluation (c'td)

- 2 out of 3 erroneous (exhaustive) simulation runs did not terminate
- the optimum architectural model terminated with lower system states and transitions than one of the functional models

  it was close to the states and transitions of the (simple) functional models

  this proves that a more refined system does **not** necessarily cause state explosion due to performance enhancement
- filtering of system states / transitions may hide (serious) problems

❒ conclusion

- performance optimisation for the target system is a critical source of bugs

  (which may not be detected by SDL / SDL tools in the current shape)
- limiting verification to a (simple) functional model only and extending / modifying this model later towards the real functionality **invalidates** the previous results

Consequences

- **consequence 1:**

  - behavioural, functional and performance aspects need to be considered **already** during simulation
  - for comparison and cross-checking, the response from the real system should **always** be available

- **consequence 2:**

  - the **final** system needs to be subject of validation / exhaustive simulation
  - it is not sufficient and dissatisfying to "play" with simplified models
  - system states need to be sufficiently low to succeed with exhaustive simulation

Performance and SDL

❒ What is meant by "performance extension of SDL" ?

- **representative** performance modelling
- **not meant:** stochastic modelling of arrival times, queuing times, latencies, ...

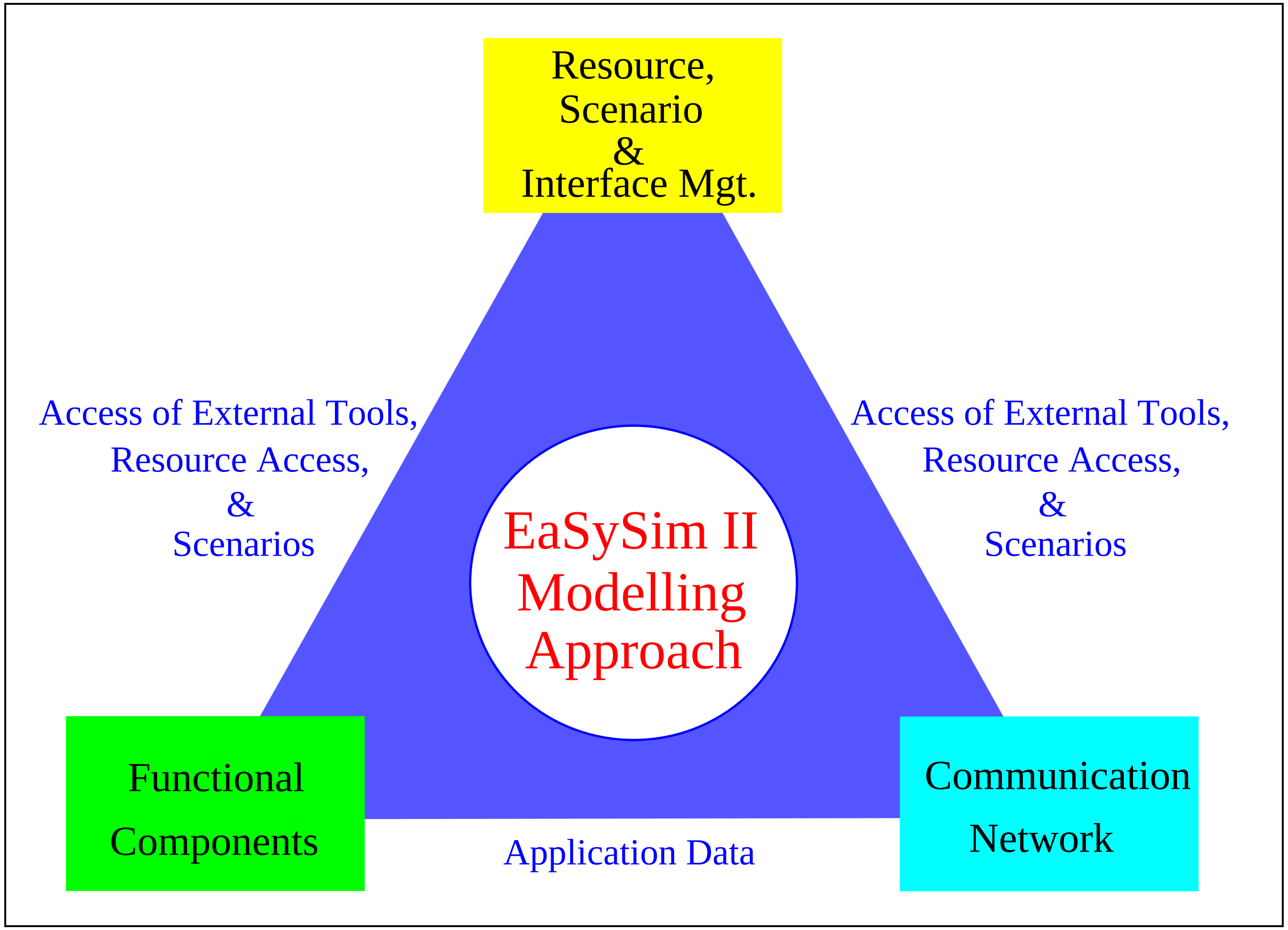
❒ Why?

- phases between signals will impact the verification result
- performance may change the order of signals in a correlated manner
  not possible by pure functional or stochastic performance modelling
- on different physical channels different propagation times may occur
- such systematic impact may only randomly hit by stochastic performance modelling

❒ What for which purpose?

- stochastic performance modelling for estimation of throughput
- accurate performance modelling for correct system validation and for minimisation of system states and transitions, and for calculation of throughput

The EaSySim II Δ-Approach



EaSySim II (1)

❐ History

- EaSySim II is based on the experience obtained by the previous ESA/ESTEC studies OMBSIM and DDV

❐ Status

- implementation of an on-board space data management system (case study)
  from specification to design and target environment

- implementation of the generic approach for a distributed system (air traffic control)
  re-entrant SDL processes

- realisation of a fault-tolerant commercial data acquisition system coupling of system components via Ethernet/ISDN, and coupling with Oracle DB
  heterogenous, distributed environment

EaSySim II (2)

❒ Features

- extends SDL and ObjectGEODE tool towards performance validation
- is based on extensions written in SDL and C
- allows for communication between SDL and "foreign", self-standing software
- allows to deal with on-line (re)configuration of fault-tolerant systems
- allows to reconfigure a system without recompilation
- provides a modelling philosophy which optimises system validation
- provides the corresponding environment
- provides templates for building SDL processes
- provides error detection mechanisms and related software
- provides software to evaluate performance loads
- allows for generic modelling: one SDL process + n data sets
- enforces reuse

Conclusions (1)

- **risk reduction and higher quality due to**
  - powerful verification and code generation capabilities of SDL and ObjectGEODE
  - enhancement by performance matters towards complete and early system validation
  - continuous verification and cross-checking capabilities over the life cycle
  - automated transition: 15 min for installation, verification and execution:
    simulation (UNIX),
    code generation (UNIX,VxWorks),
    execution (UNIX,VxWorks)

Conclusions (2)

- **higher efficiency at higher quality**
  - EaSySim II: *~300 operators, 5.000 SDL LOC (net), 25.000 C LOC (net)*
  - ATC study (simulation only)

    ~4.500 SDL LOC (net), 250 man hours, 40 C LOC per man hour
  - commercial fault-tolerant data acquisition system

    ~ 10.000 SDL LOC (net, application)
    (34.000 LOC in pr-file incl. EaSySim II)
    yielding ~20.000 C LOC

    ~ 20.000 C LOC (net) in addition (without EaSySim II and database)

    ~ 40.000 C LOC (net) in total, existing now

    ~ 2500 man hours for completion (estimation)

    ~ 16 C LOC per man hour
    not visible by the figures: degree of reuse in SDL and C by EaSySim II
    organisation principles