Methodology of Communication Protocols Development: from Requirements to Implementation

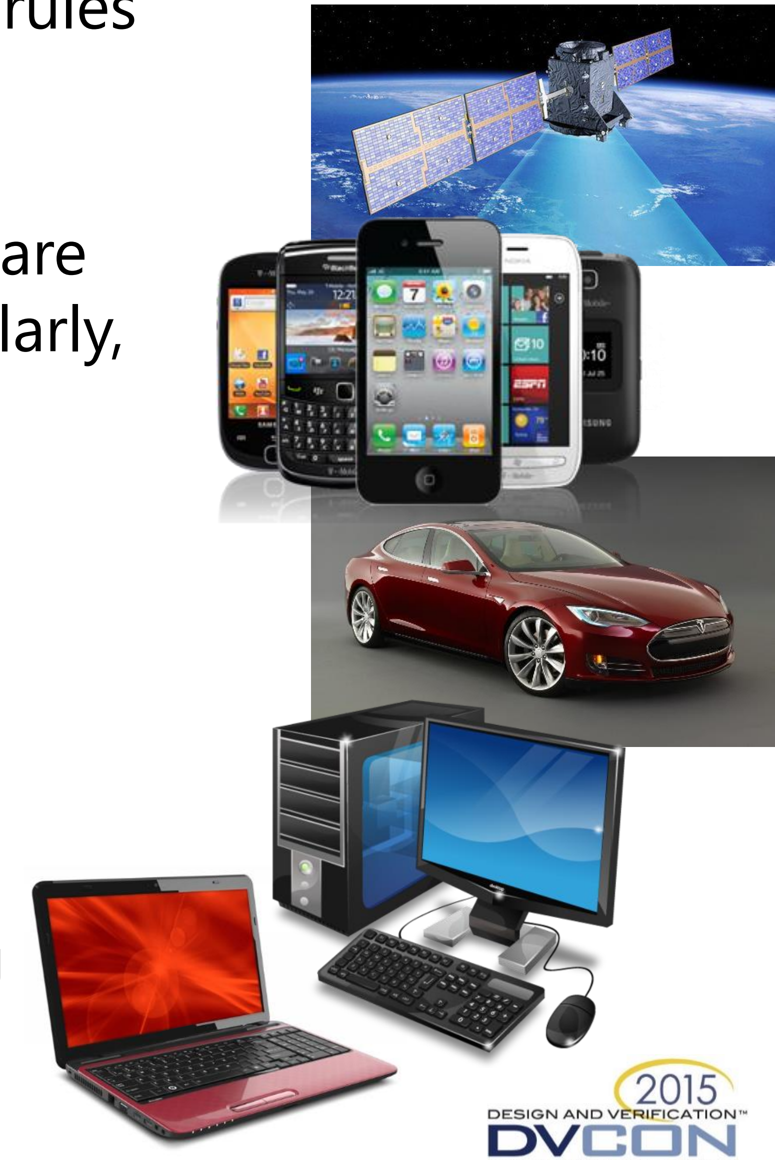
Irina Lavrovskaya, Valentin Olenev

{irina.lavrovskaya, valentin.olenev}@guap.ru

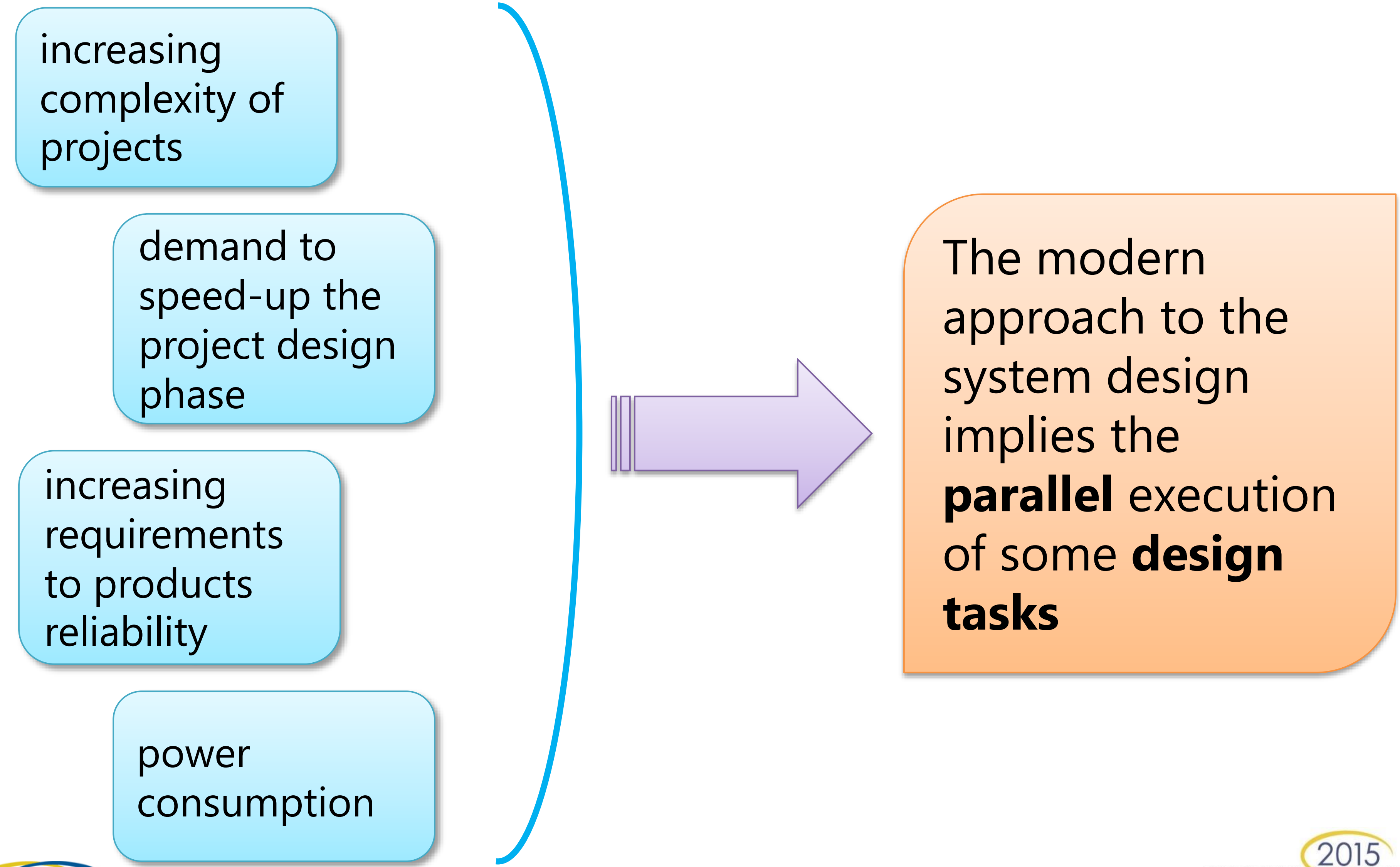
COMMUNICATION PROTOCOLS DEVELOPMENT

Communication Protocols Development

- **Communication protocol** is a set of rules for the order in which messages of particular types are exchanged
- Nowadays communication protocols are widely used in different areas. Particularly, in the following areas:
  - Space and aircraft industry;
  - Vehicle systems;
  - Mobile industry (USB, UniPro, etc.);
  - Computing (TCP/IP, etc.);
  - Etc.
- The developed protocol should be **precisely investigated before being implemented**



Protocol Design Difficulties



increasing complexity of projects

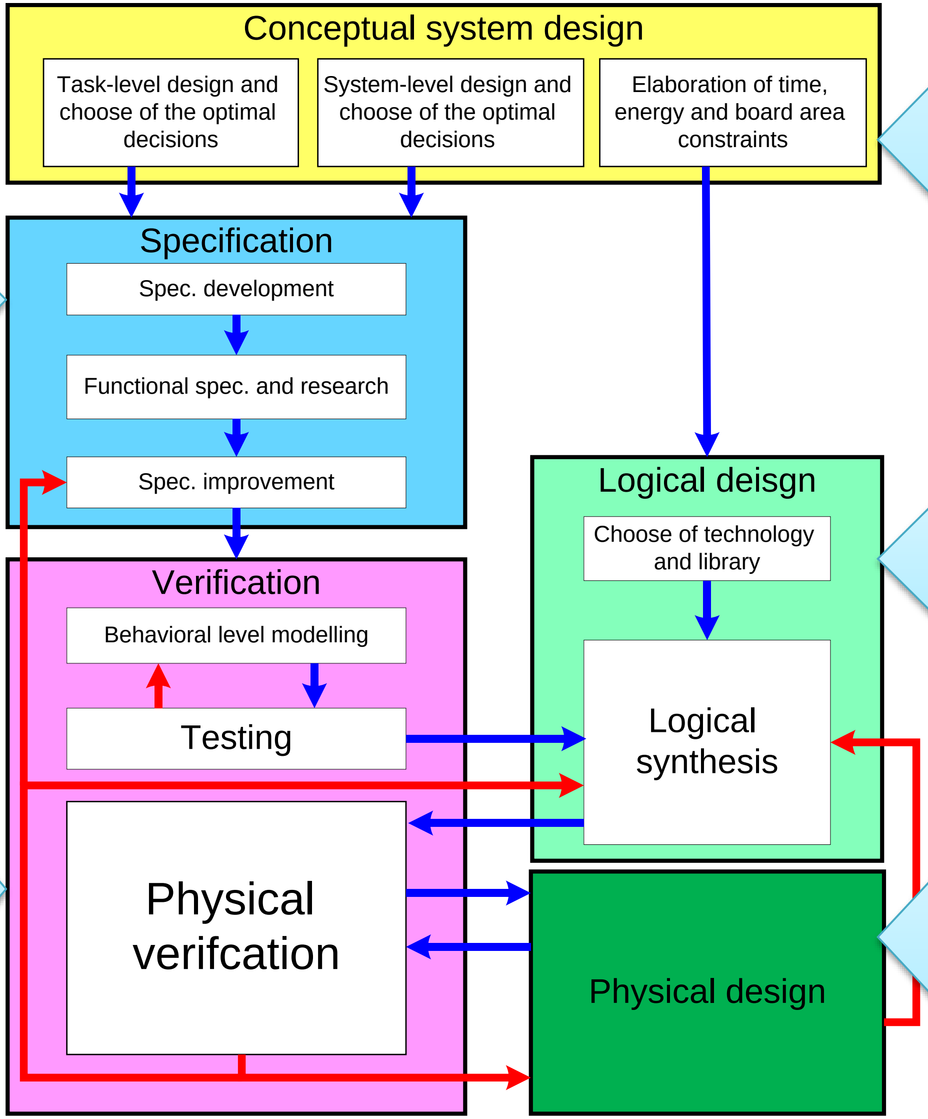
demand to speed-up the project design phase

increasing requirements to products reliability

power consumption

The modern approach to the system design implies the **parallel** execution of some **design tasks**

System General Design Flow:



Conceptual system design

Task-level design and choose of the optimal decisions

System-level design and choose of the optimal decisions

Elaboration of time, energy and board area constraints

Specification

Spec. development

Functional spec. and research

Spec. improvement

Verification

Behavioral level modelling

Testing

Physical verifcation

Logical deisgn

Choose of technology and library

Logical synthesis

Physical design

Collecting requirements for the target system, evaluation of the main system description, analysis of the system mechanisms and development of the first specification draft

Getting the final version of the system specification and the system model in a high-level language (usually in C/C++, SDL)

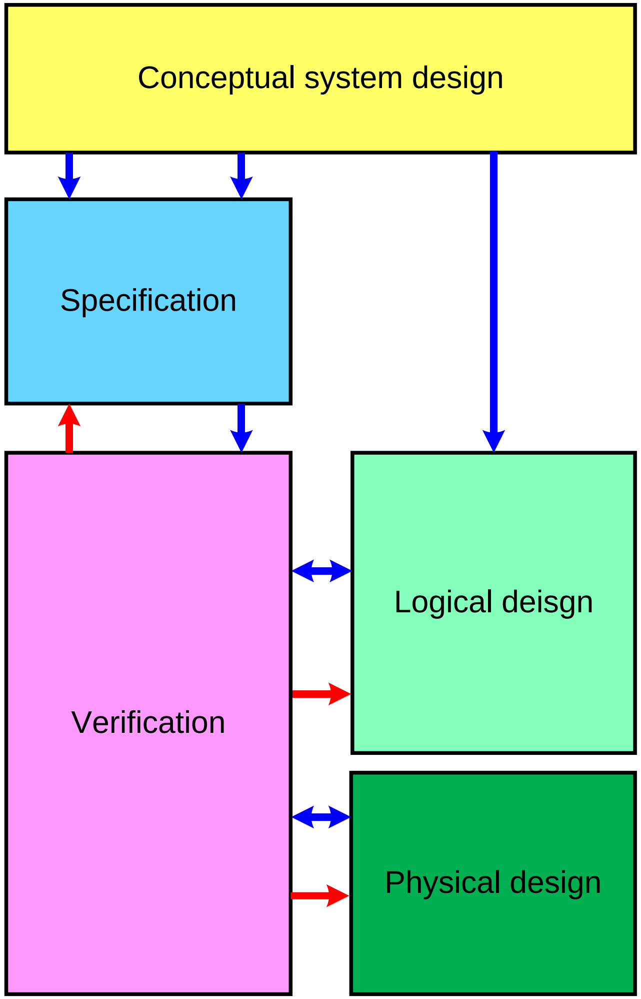
Translation of the executable project specification to the register level (in Verilog/VHDL) and further at the gate level

Verification of the design decisions on conformity to the specification and other requirements

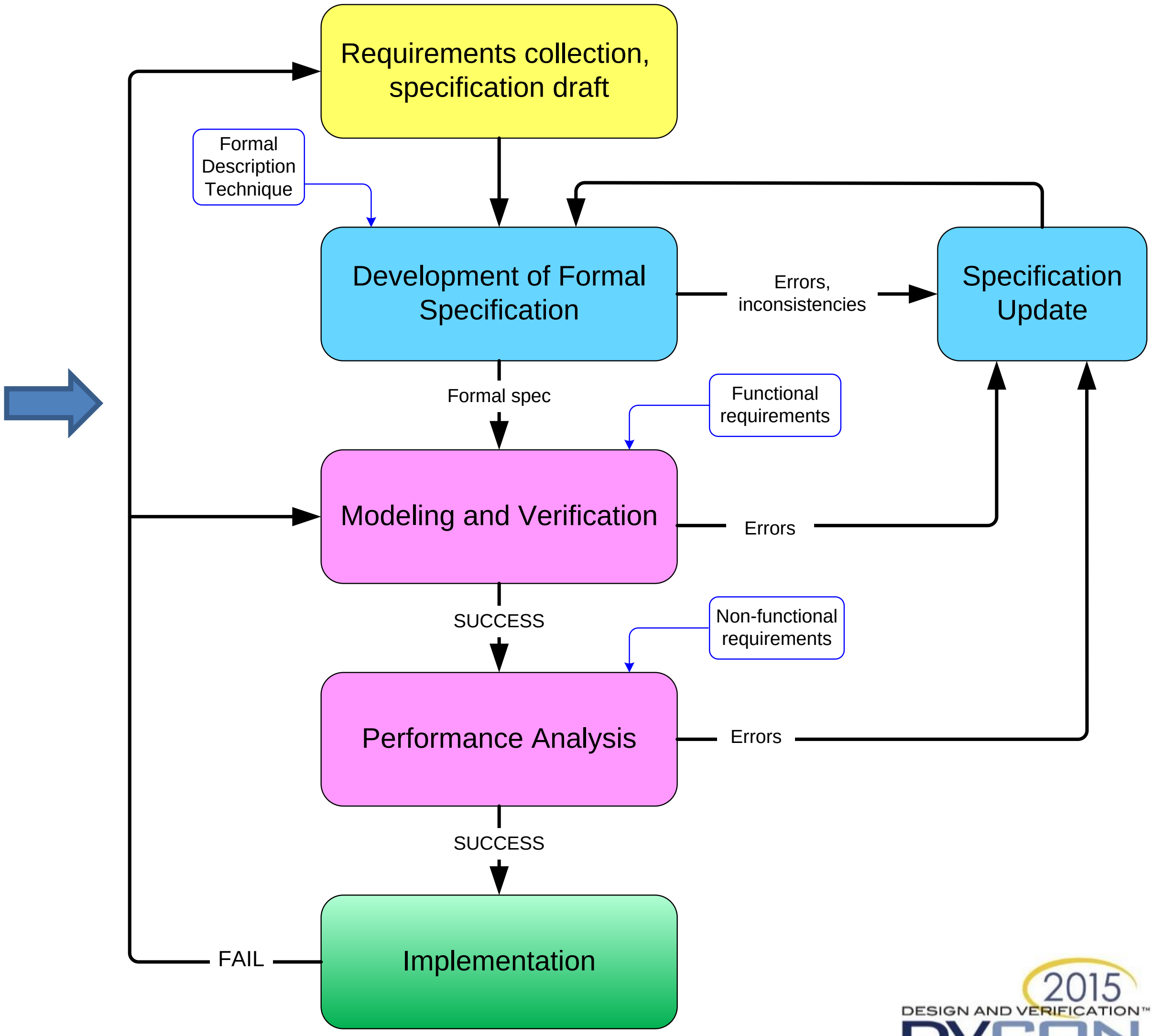
This stage begins from the selection of technological and library basis and it is completed when everything is ready for the final product production

What do we Cover by our Methodology?

Generalised design flow



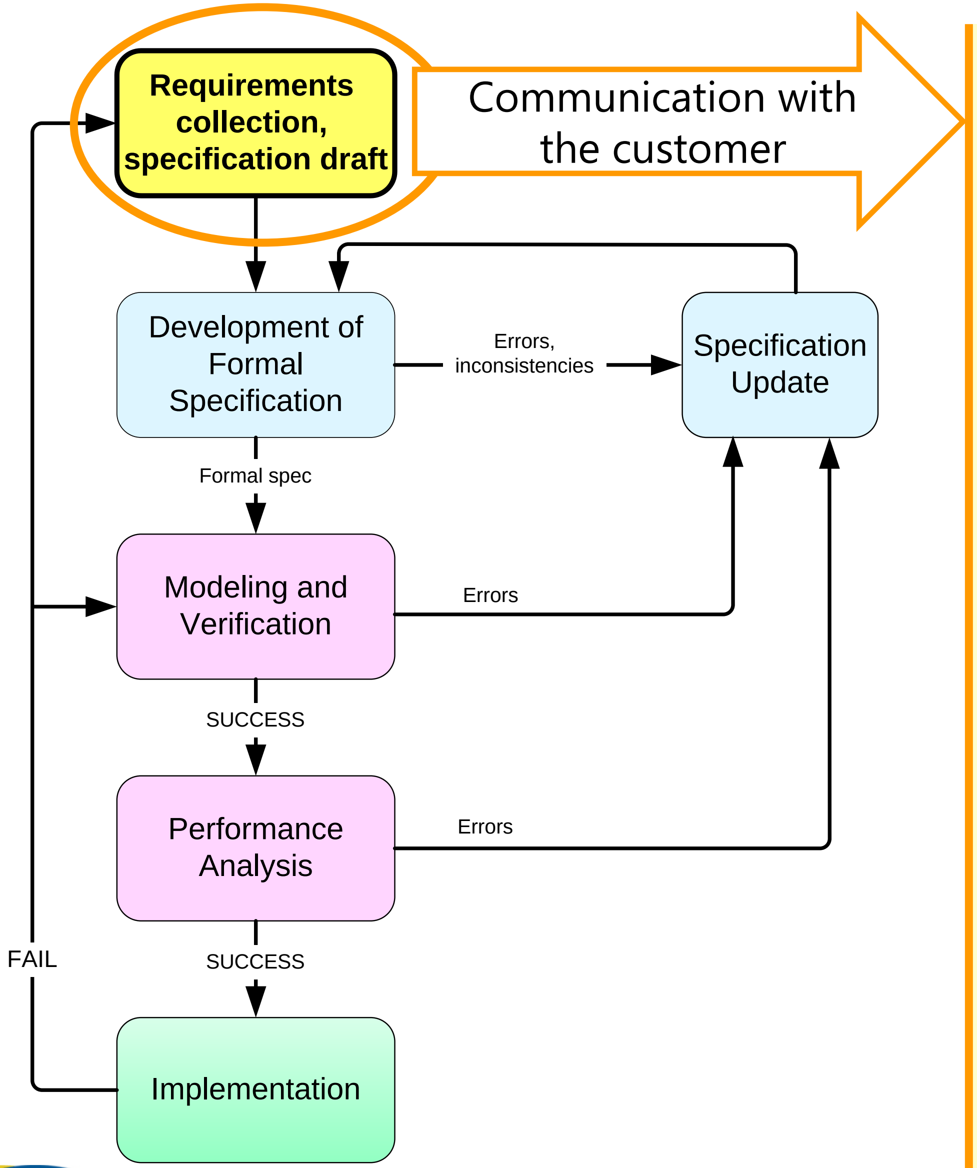
Our approach



Requirements collection, specification draft
Formal Description Technique
Development of Formal Specification
Errors, inconsistencies
Specification Update
Formal spec
Functional requirements
Modeling and Verification
Errors
SUCCESS
Non-functional requirements
Performance Analysis
Errors
SUCCESS
FAIL
Implementation

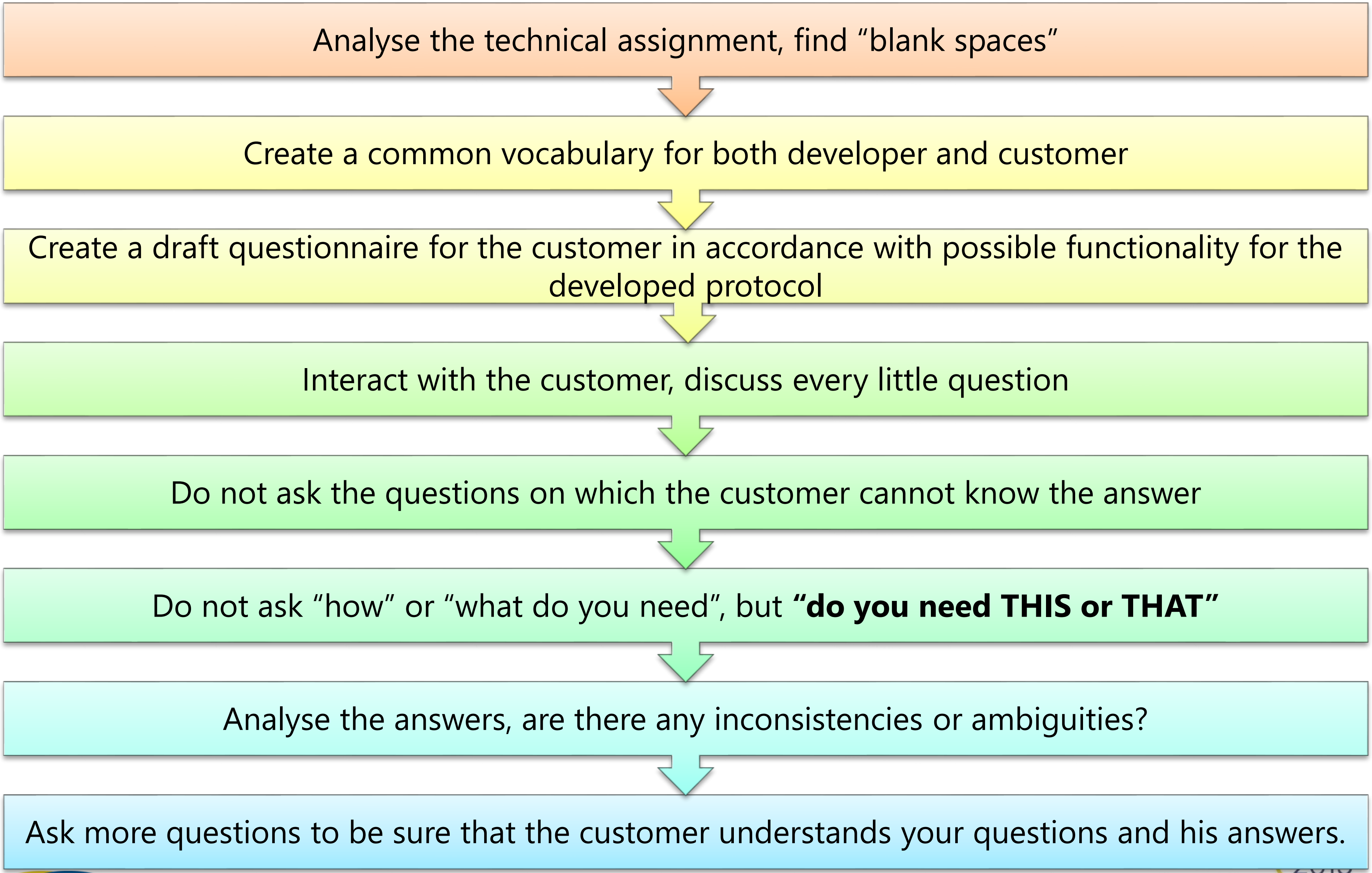
Collecting Requirements

Main Problems to Resolve


Requirements collection, specification draft
Communication with the customer
Development of Formal Specification
Errors, inconsistencies
Specification Update
Formal spec
Modeling and Verification
Errors
SUCCESS
Performance Analysis
Errors
FAIL
SUCCESS
Implementation

- Different terminology;
- Unclear requirements;
- Customer who:
  - does not know what they want;
  - wants more than they need;
  - wants to get smth absolutely new and keep all adjacent soft/hardware unchanged;
- Some parameters and requirements should be held in confidence, e.g. in space industry companies.

Main Principles of Collecting Requirements



Analyse the technical assignment, find “blank spaces”

Create a common vocabulary for both developer and customer

Create a draft questionnaire for the customer in accordance with possible functionality for the developed protocol

Interact with the customer, discuss every little question

Do not ask the questions on which the customer cannot know the answer

Do not ask “how” or “what do you need”, but **“do you need THIS or THAT”**

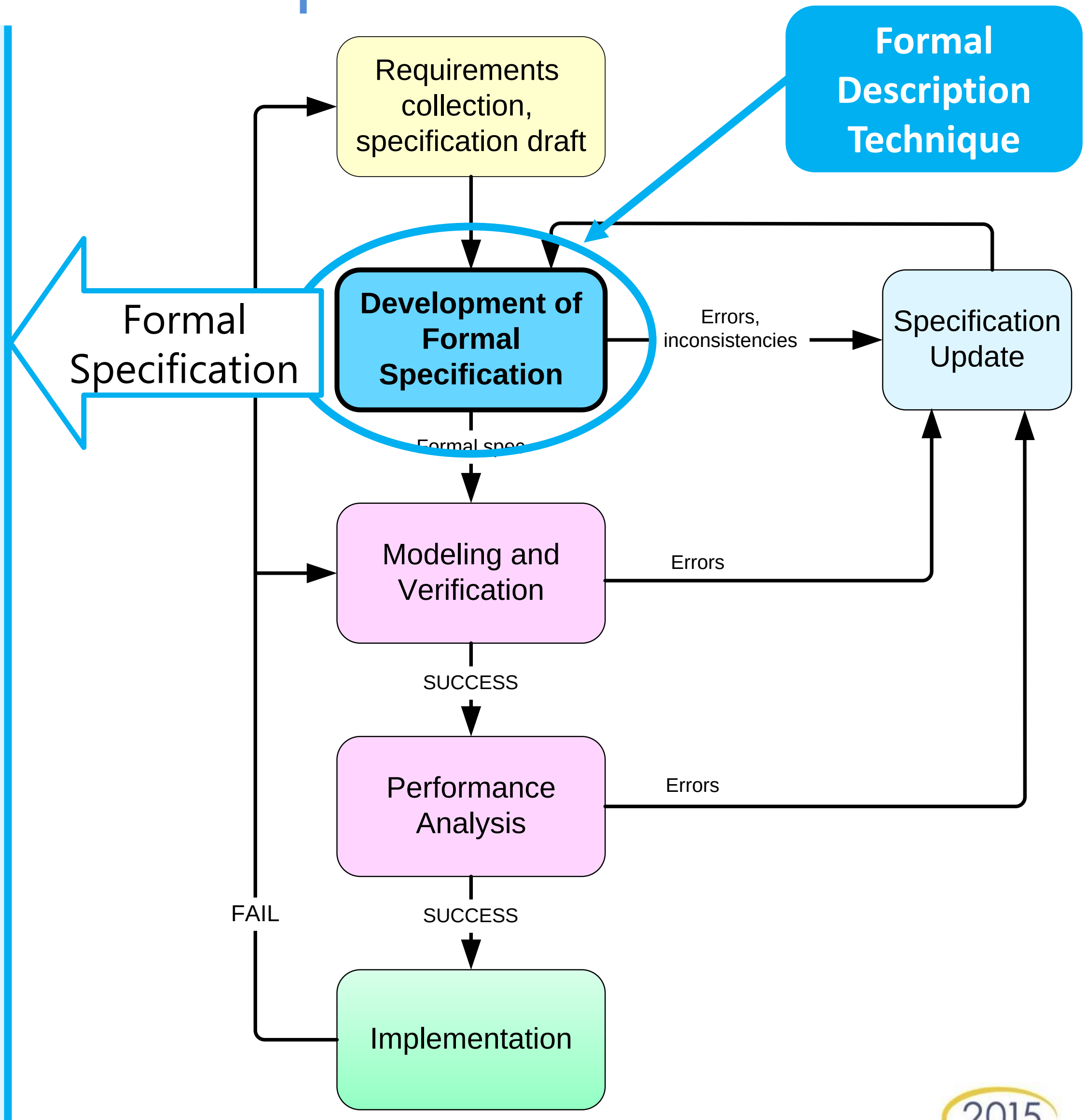
Analyse the answers, are there any inconsistencies or ambiguities?

Ask more questions to be sure that the customer understands your questions and his answers.

Specification and Description Language Formal Spec

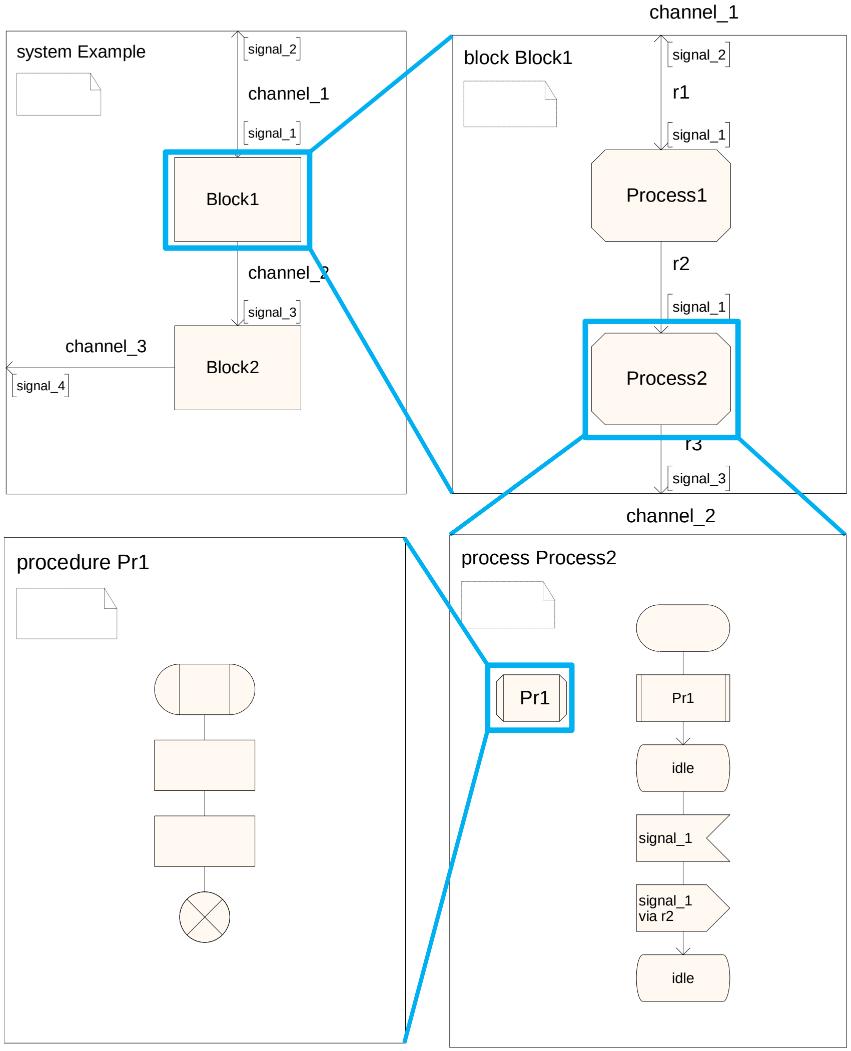
Formal Protocol Specification

- **The objective of development**: the target model describes all mechanisms, interactions and functionality which are stated in the specification.
- There is a set of FDT: Spin, Estelle, LOTOS, Petri Nets, SDL, etc.
- In our methodology we widely use **Specification and Description Language (SDL)** for protocol specification
- **Results of this stage**:
  - consistent readable textual specification
  - formalised graphical specification in SDL is produced which can be used as a reference for the textual spec



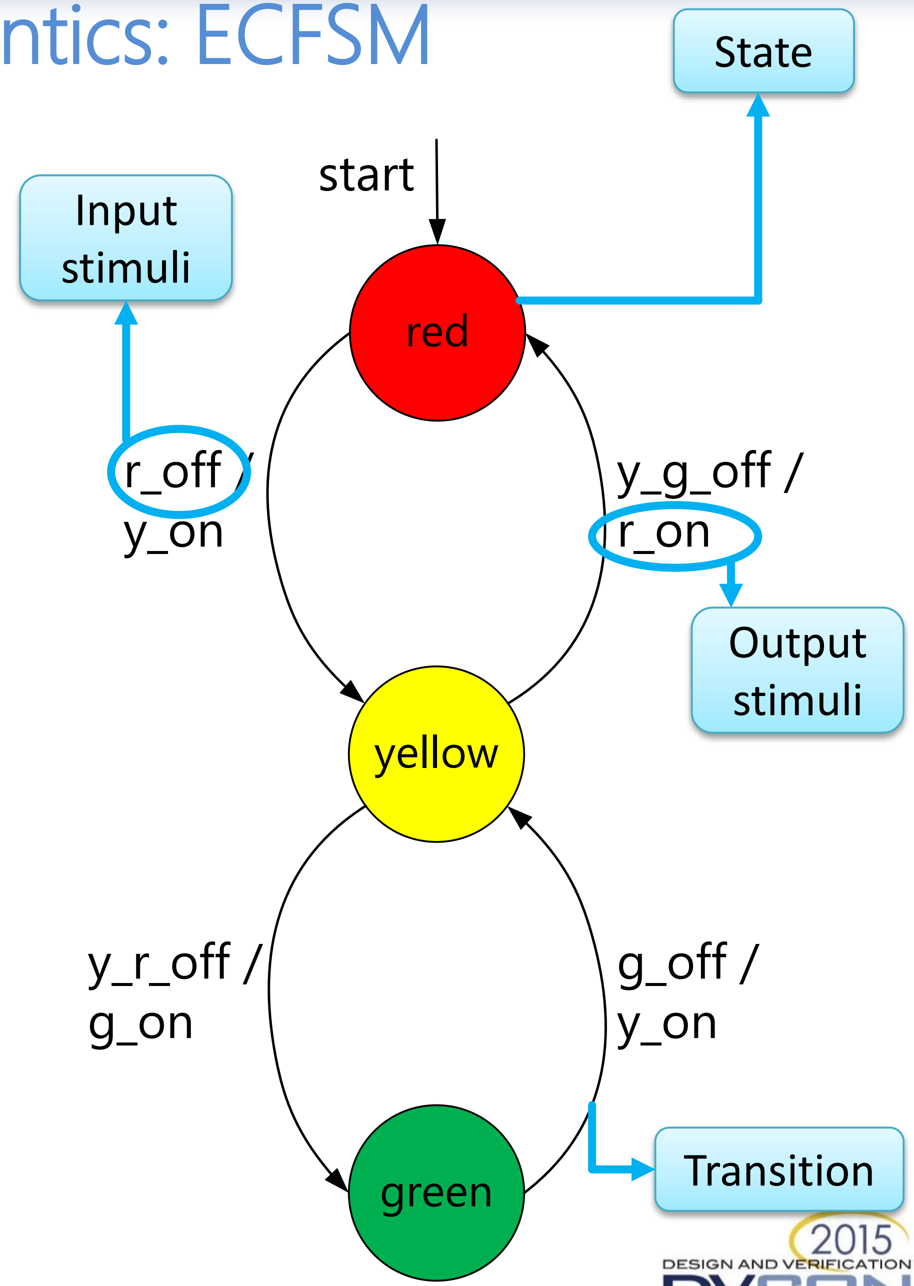
Specification and Description Language (SDL)

- Standardized between 1976 and 1992 by ITU-T
- High-level general-purpose graphical description language for event-driven, real-time and communicating systems
- SDL provides two representations:
  - Graphical (SDL-GR)
  - Textual (SDL-PR)
- Application fields:
  - telecommunication systems
  - protocols
- Provides strong structuring facilities which give an ability to describe systems of all kinds of difficulties



Formal Semantics: ECFSM

- The system described by an SDL specification represents the Extending Communicating Finite State Machine (ECFSM)
  - It consists of a set of concurrent processes, extended with variables and data space.
  - Communication is performed by exchanging control signals on finite-length asynchronous channels. Output signals of one process can be an input signal for the another process.
  - Each process consists of a set of states. Transitions from one state to another are performed in accordance with the received signals.



Example of Protocol Stack SDL Specification

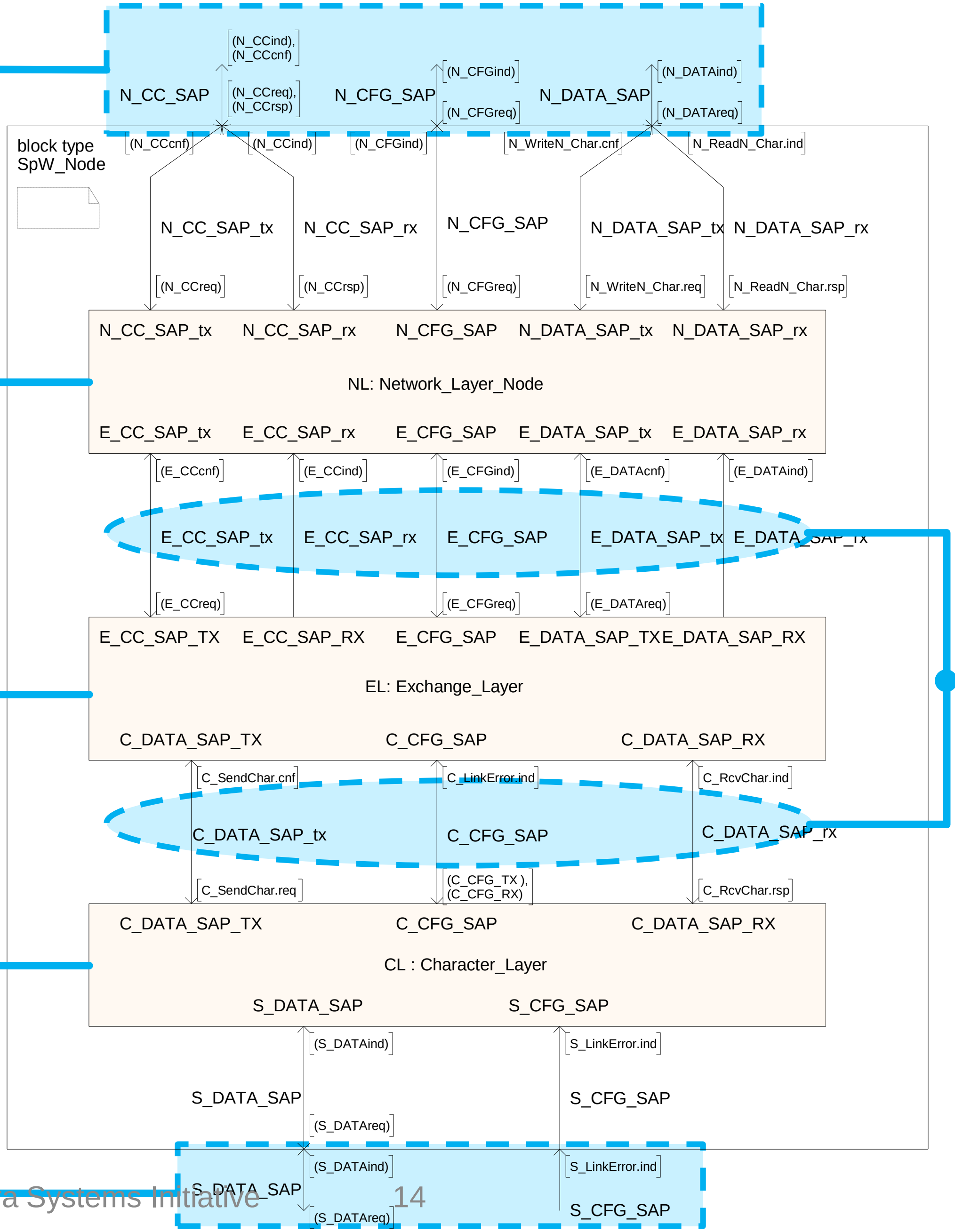
Upper Service Access Point

SpaceWire protocol layer blocks

Bottom Service Access Point

SDL specification of the SpaceWire protocol stack

Internal interactions via channels



Modeling, Verification, Performance Analysis

Protocol Modeling

Main goals of modeling

- Verification of the protocol functional properties;
- Investigation of compatibility and correctness of algorithms and methods deployed in the specification;
- Investigation of protocol operation in case of error occurrence while data transmission.

Modeling and investigation directions

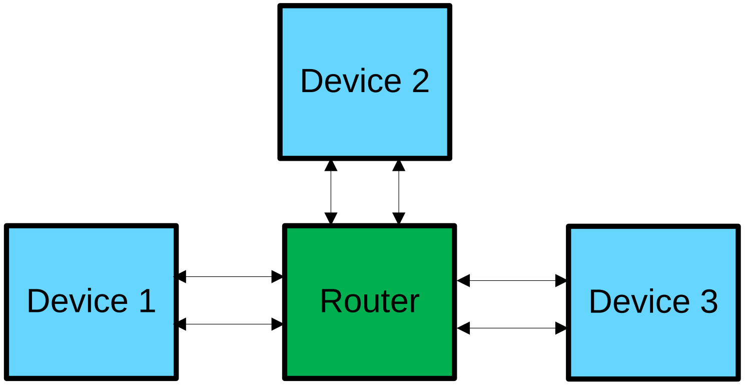
- Specification and Description Language;
- SystemC modeling;
- C++ reference code;
- SDL/SystemC joint model.

Basic approaches

- Protocol stack modeling;
- Network modeling.

Approaches for Protocol Modeling (1/2)

Network modeling



Goals:

- check the data transmission
- check the routing correctness

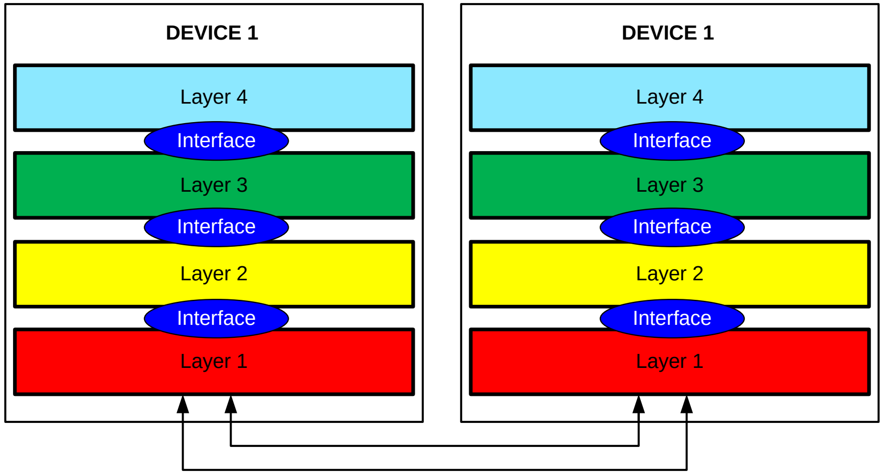
Benefits:

- The real interest represents here the mechanisms of devices communications in the network which is the key issue for the performance analysis

We cannot consider:

- the protocol layers
- the interaction between protocol layers
- the forming of packets
- device's operation with applications

Protocol stack modeling



Goals:

- check the presence of errors in specification
- check the packets generation
- check all internal mechanisms

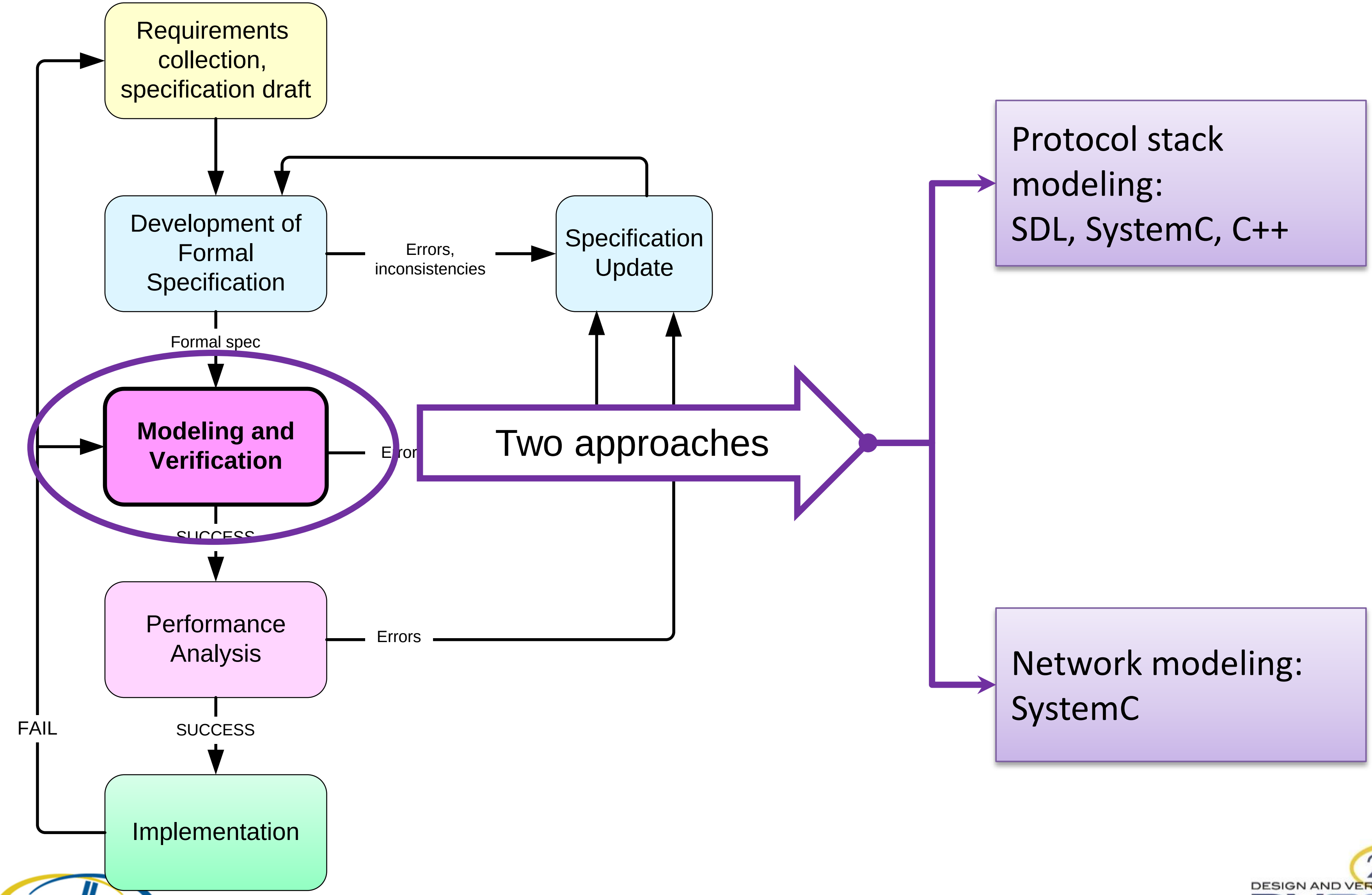
Benefits:

- The set of modules breaks into the layers forming hierarchy;
- Every layer communicates only with directly adjoining layers.

We cannot consider:

- Interaction of devices in a network

Approaches for Protocol Modeling (2/2)



Requirements collection, specification draft

Development of Formal Specification

Errors, inconsistencies

Specification Update

Formal spec

Modeling and Verification

Error

Two approaches

SUCCESS

Performance Analysis

Errors

FAIL

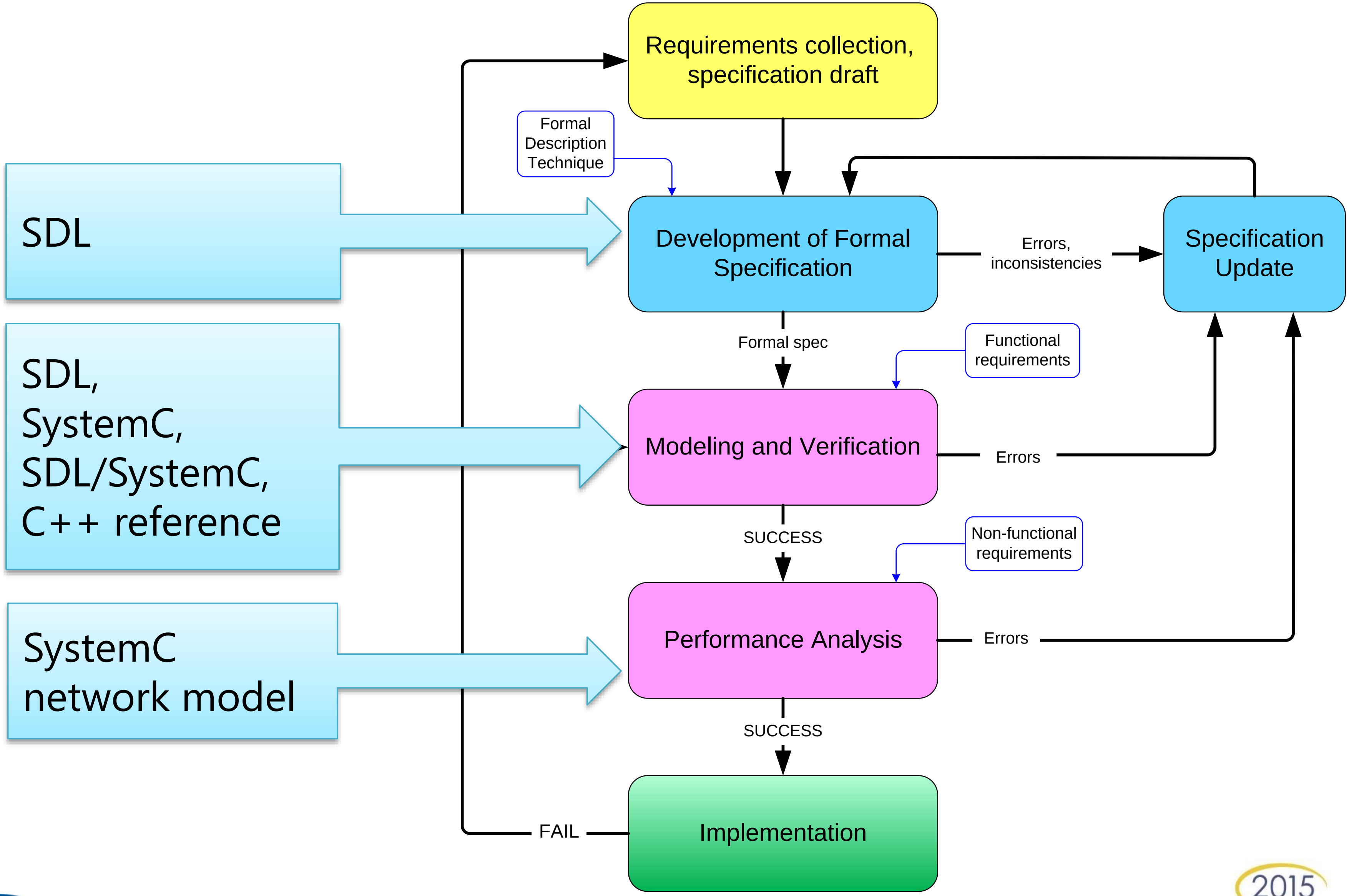
SUCCESS

Implementation

Protocol stack modeling:
SDL, SystemC, C++

Network modeling:
SystemC

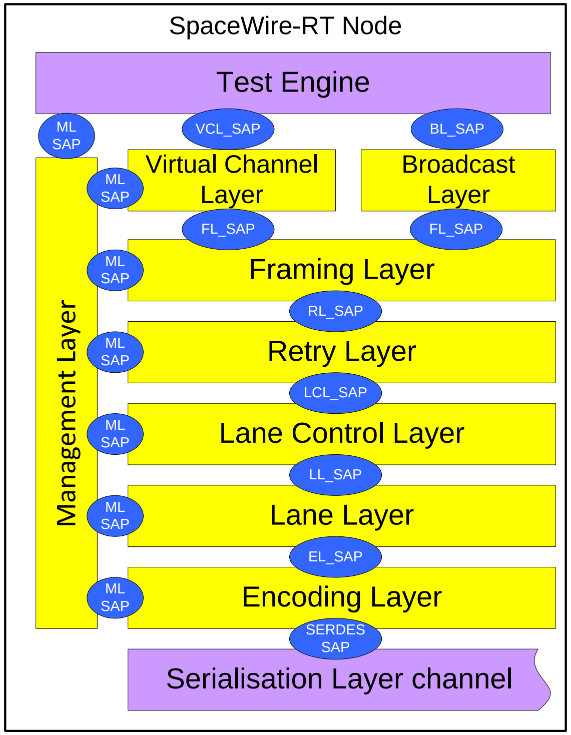
Mapping of Modeling Directions to Design Flow Stages

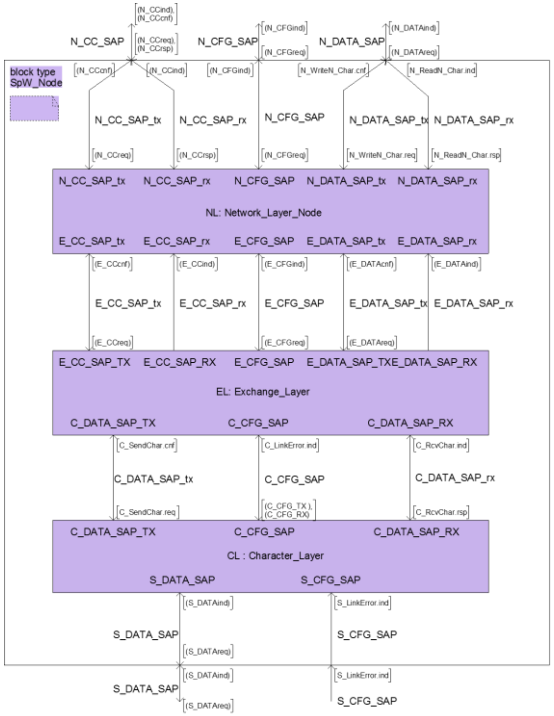

Requirements collection, specification draft
Formal Description Technique
SDL
Development of Formal Specification
Errors, inconsistencies
Specification Update
Formal spec
Functional requirements
SDL, SystemC, SDL/SystemC, C++ reference
Modeling and Verification
Errors
SUCCESS
Non-functional requirements
SystemC network model
Performance Analysis
Errors
SUCCESS
FAIL
Implementation

SDL Protocol Stack Modeling

SDL Protocol Stack Modeling

- SDL is the most reasonable solution for modeling and validation on per layer basis.
- SDL model formally describes all mechanisms, interactions and functionality stated in the specification.
- Generally, the simulated system consists of two nodes which communicate via a model of a link.



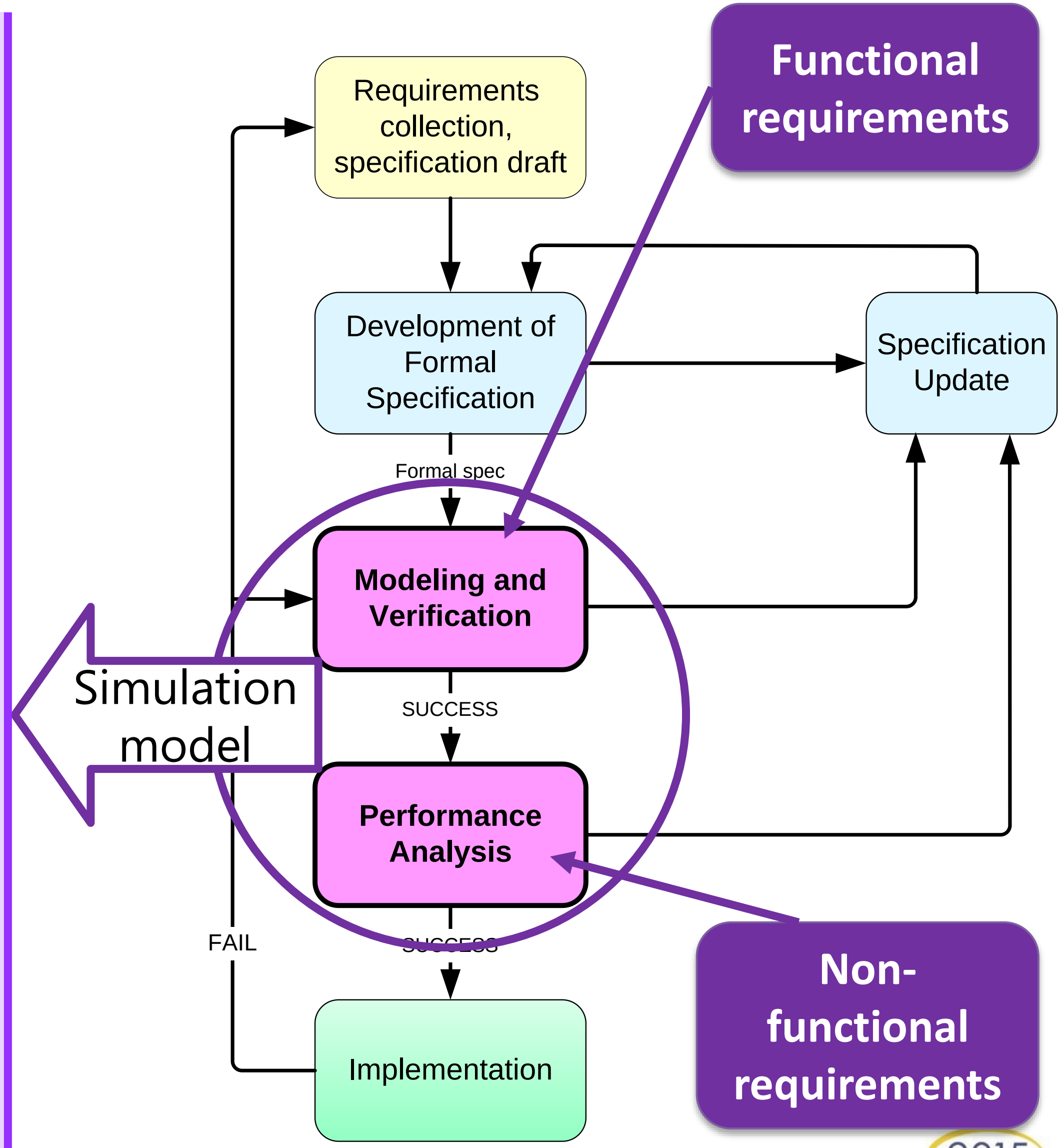


- Such kind of modeling gives an ability to:
  - check and verify all internal mechanisms,
  - validate the consistency of the specification and
  - check functional requirements that were defined for the protocol.
- Furthermore, such SDL models can be used as a part of a tester.

SystemC Network Modeling

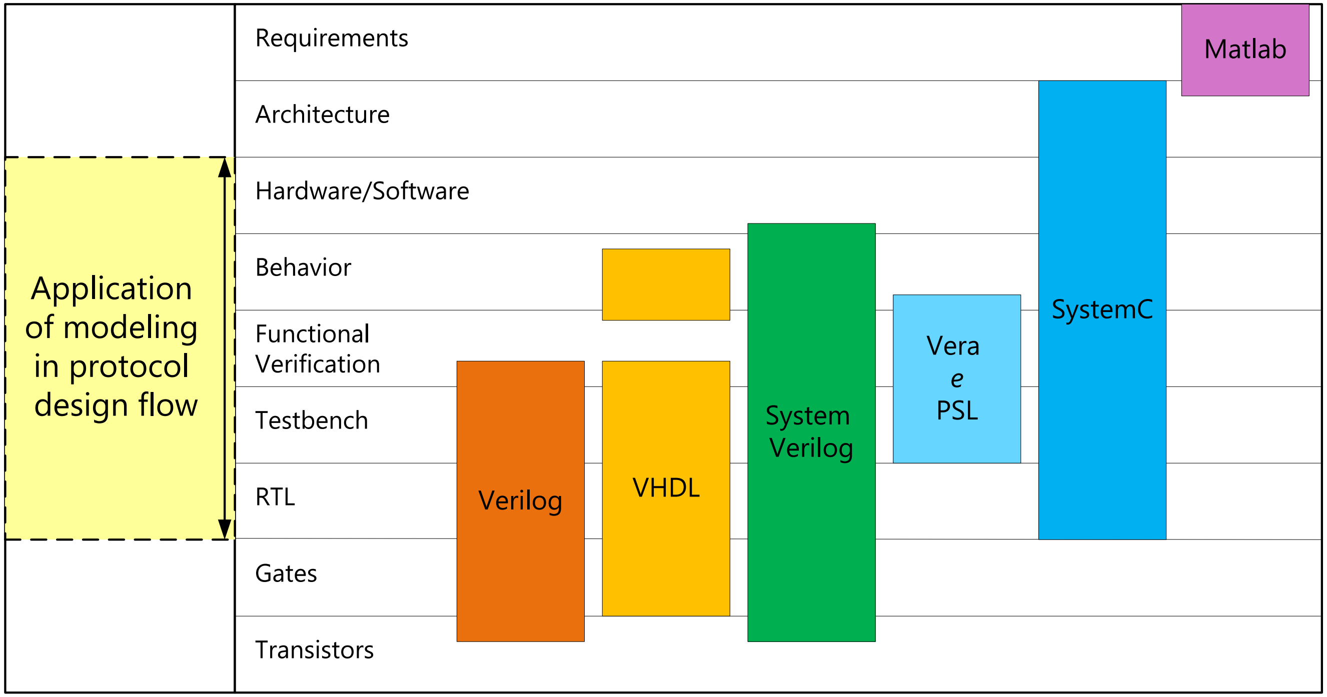
SystemC

- SystemC is a C++ library for modeling of embedded systems and communication protocols
- **The objective of development**:
  - check the communication protocol operation over the network,
  - test the network configuration,
  - networking features and conduct the performance analysis
- In our methodology we widely use **SystemC** for network modeling and performance analysis
- **Results of this stage:**
  - the final version of the system specification
  - the system model in a high-level language

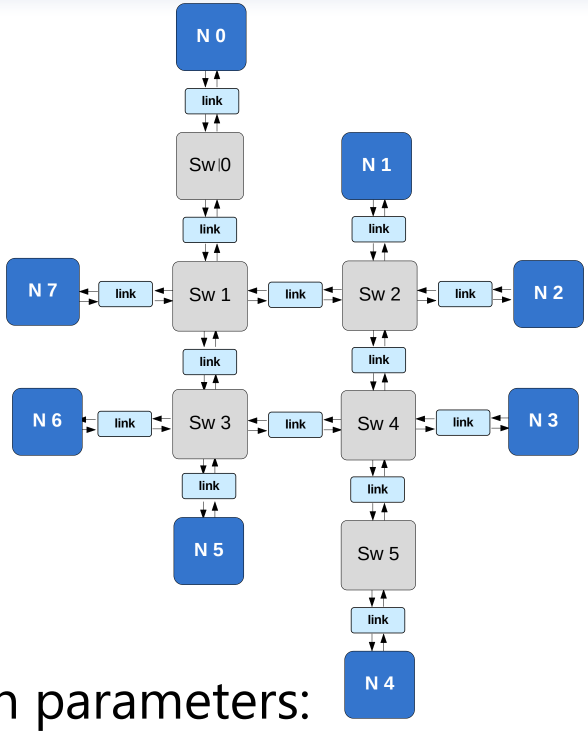

Functional requirements
Requirements collection, specification draft
Development of Formal Specification
Specification Update
Formal spec
Modeling and Verification
Simulation model
SUCCESS
Performance Analysis
FAIL
SUCCESS
Implementation
Non-functional requirements

SystemC Advantages

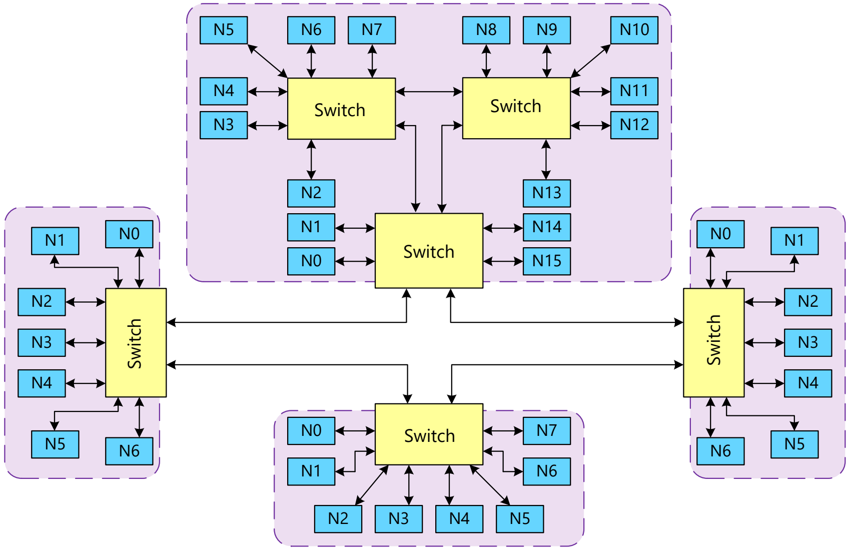
- SystemC uses
  - such primitives as channels, interfaces and methods,
  - it gives high flexibility in modeling that could be based on various computation models,
  - provides possibility to integrate and use these models in parallel.
- SystemC is C++ based and this point makes cooperation in HW/SW design easier.
- SystemC supports hardware modeling and detailing of a project to the RTL level.



Modeling of Networks



- Generally, the network model consists of the following SystemC modules:
  - Nodes implementing the communication protocol,
  - Switches or routers,
  - Traffic Generators, which operate over the nodes and give ability to launch different tests and generate test sequences.



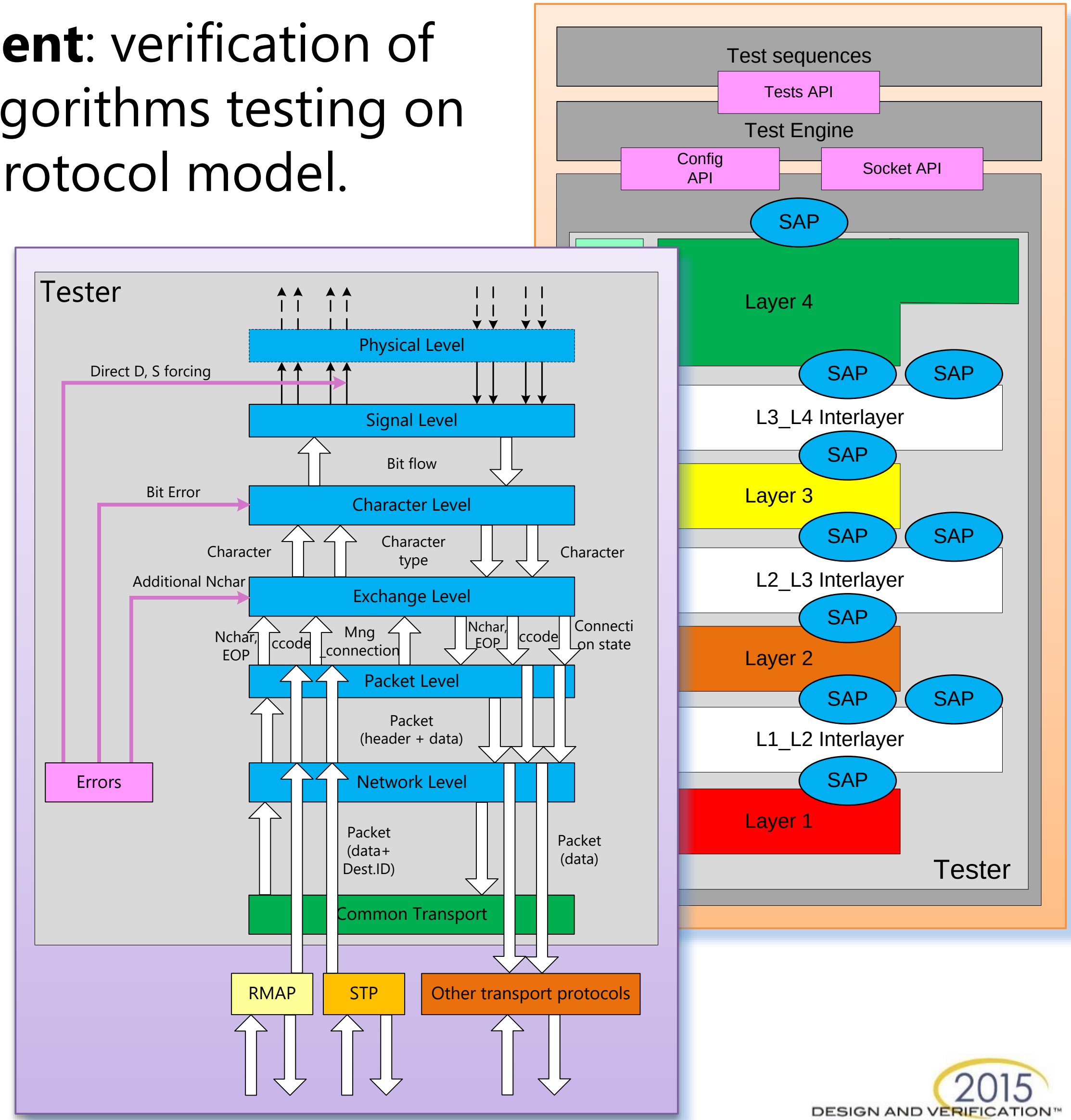
- Configuration parameters:
  - data transmission speed,
  - number of nodes and switches,
  - time delay and routing table for the switch,
  - number of ports in the switch, etc.
- Ability to simulate operation of:
  - the various numbers of devices
  - networks with different topologies.

© Accellera Systems Initiative

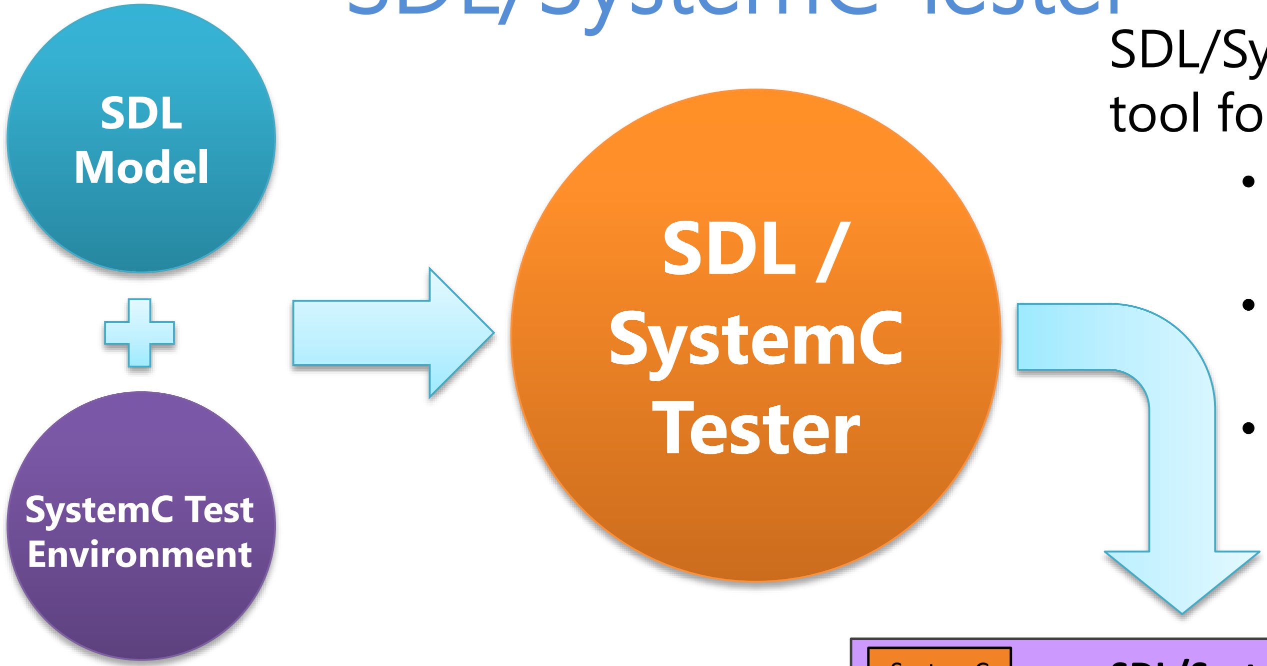
SDL/SystemC Co-Modeling. Tester

Protocol Model Tester

- **The objective of development**: verification of the protocol specification, algorithms testing on the basis of the developed protocol model.
- Such a Tester is able to:
  - validate the standard specification itself;
  - validate the model for correspondence to the specification;
  - test prototypes or boards in production;
  - certify products, verify products for conformance with the standard.



SDL/SystemC Tester

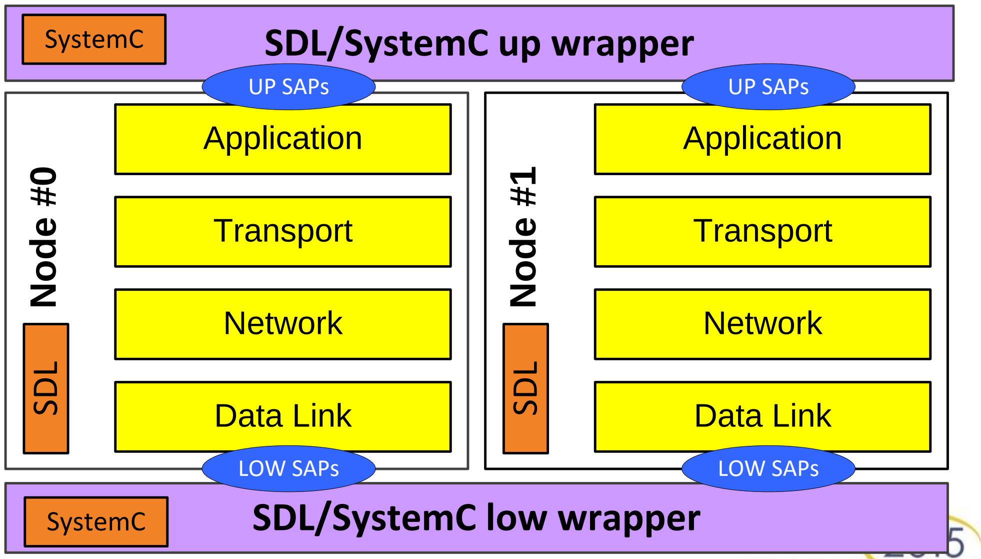


SDL/SystemC tester is a flexible tool for:

- setting different configurations;
- generating various test sequences;
- gathering statistics.

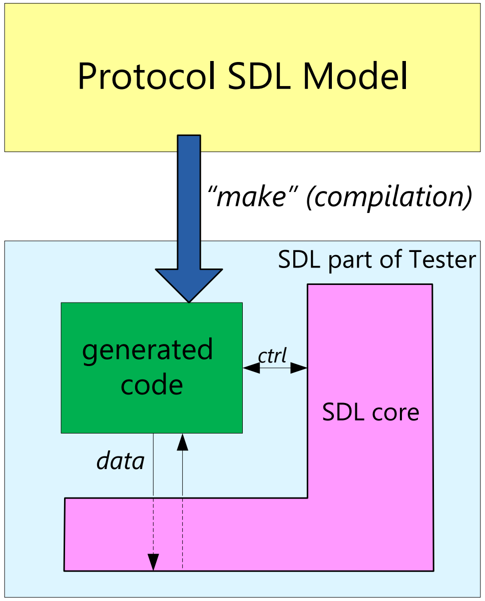
The Tester gives the following abilities:

- representation of tested layers by means of finite state machine
- access not to the whole SDL model only but also to certain layers of stack through appropriate Intermediate Blocks
- getting all necessary test results by SystemC implementation of the test environment



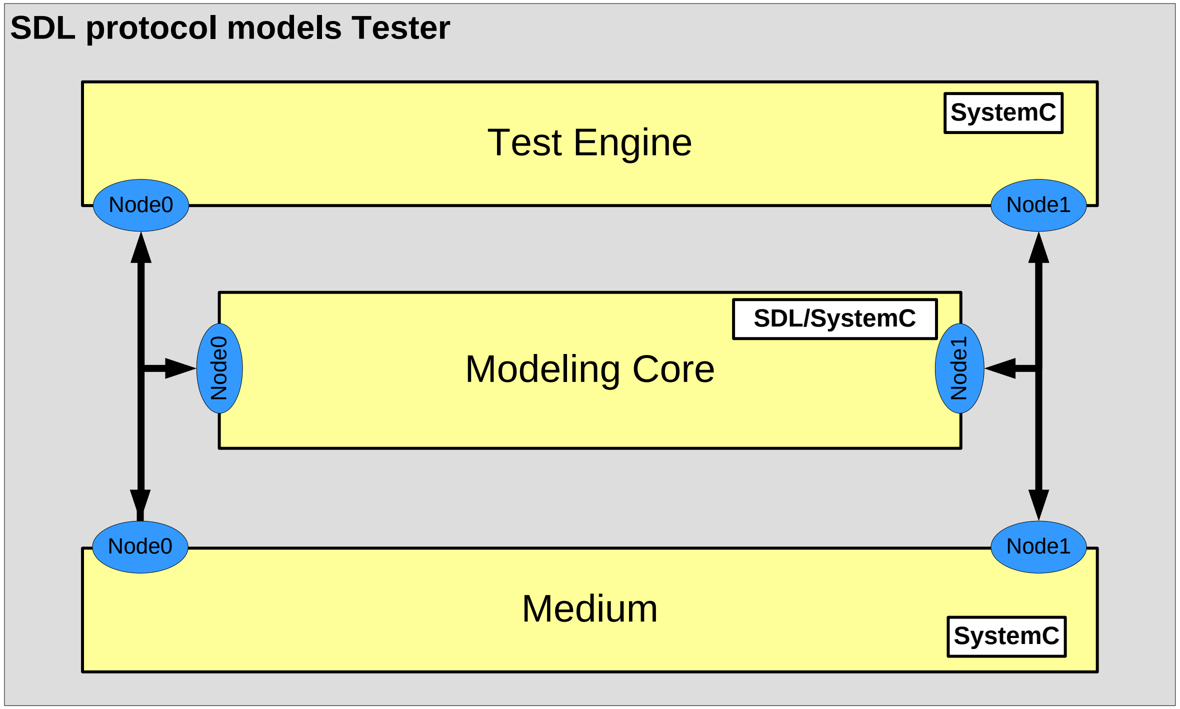
Organization of SDL and SystemC Co-Modeling

- SDL and SystemC can be combined in **one model** so as to:
  - use SDL as a basic FDT for specification, verification and performance analysis
  - use SystemC for creation of complex test sequences and to provide wide facilities to work with time
  - perform all investigations on the basis of one model of the protocol.
- **Basic structure** of SDL/SystemC co-model:
  - SystemC provides simulation core and test environment
  - SDL provides formal protocol implementation



- **General principles:**
  - SDL model is compiled into C-code, equivalent to the original model
  - Generated C-code together with the SDL core is integrated into the SystemC project

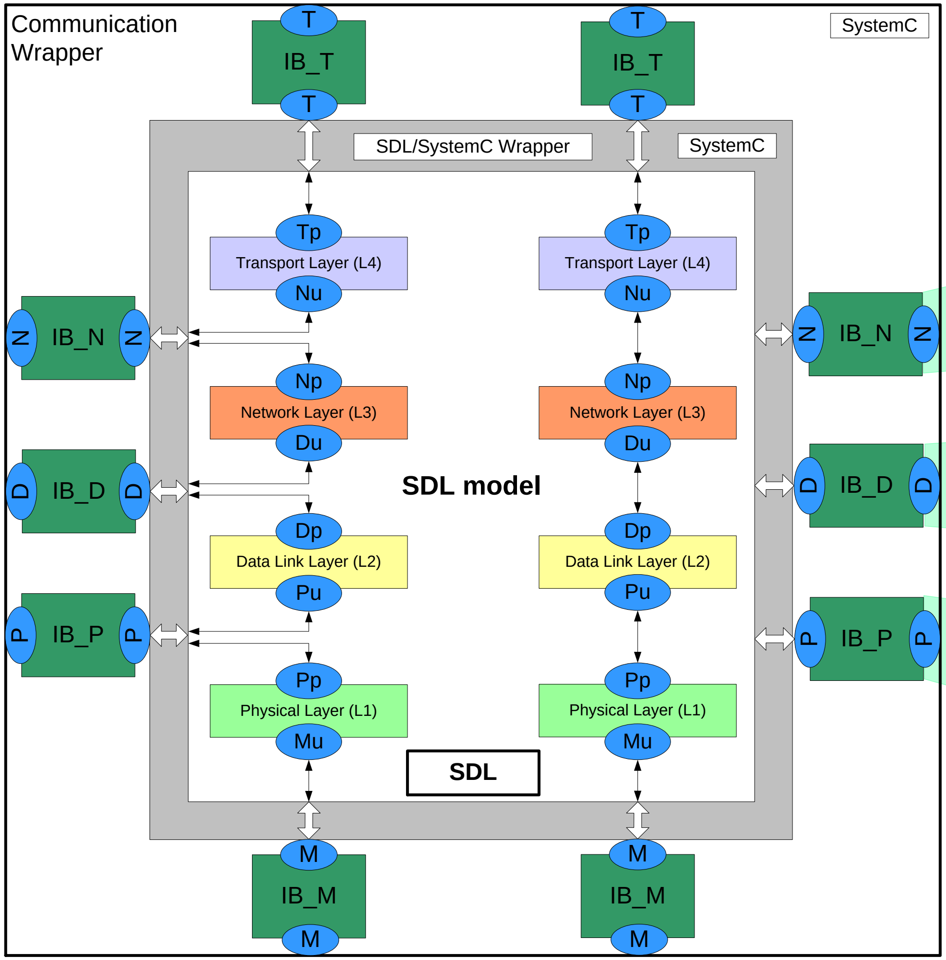
Protocol SDL Models Tester



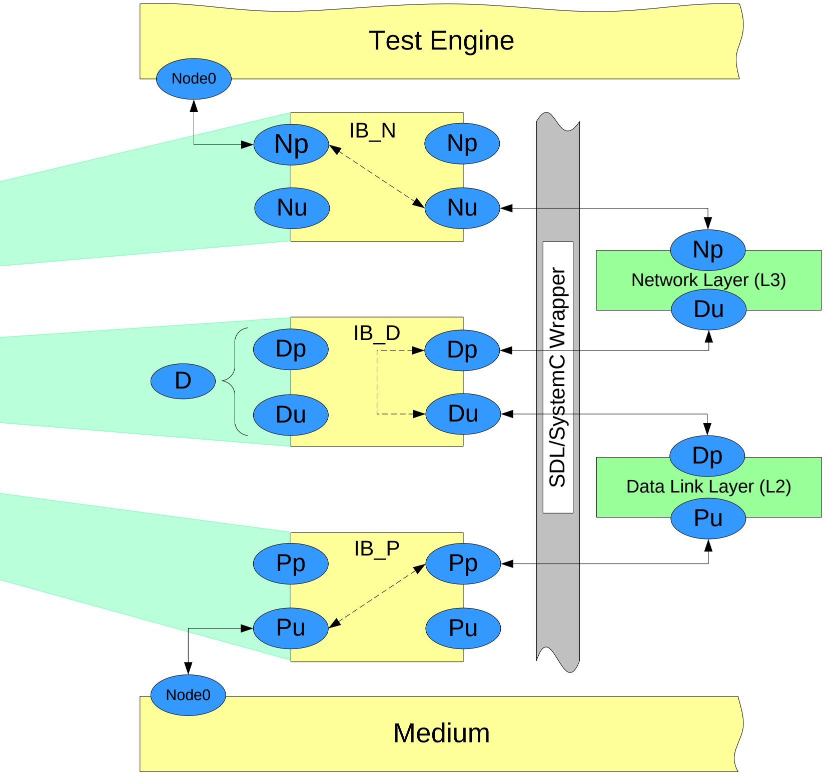
The Tester contains three parts:

- **Test Engine** – contains conformance tests and Tester control features
- **Modeling Core** – model of the tested protocols or protocol stack which are implemented according to the protocol specification in SDL
- **Medium** – ensures interconnection between tested nodes inside the Modeling Core

Modeling Core



Intermediate blocks of Communication Wrapper



🟨 - module implemented in SystemC

🟩 - module implemented in SDL

The Modeling Core consists of:

- Tested SDL model
- SDL/SystemC Wrapper
- Communication Wrapper

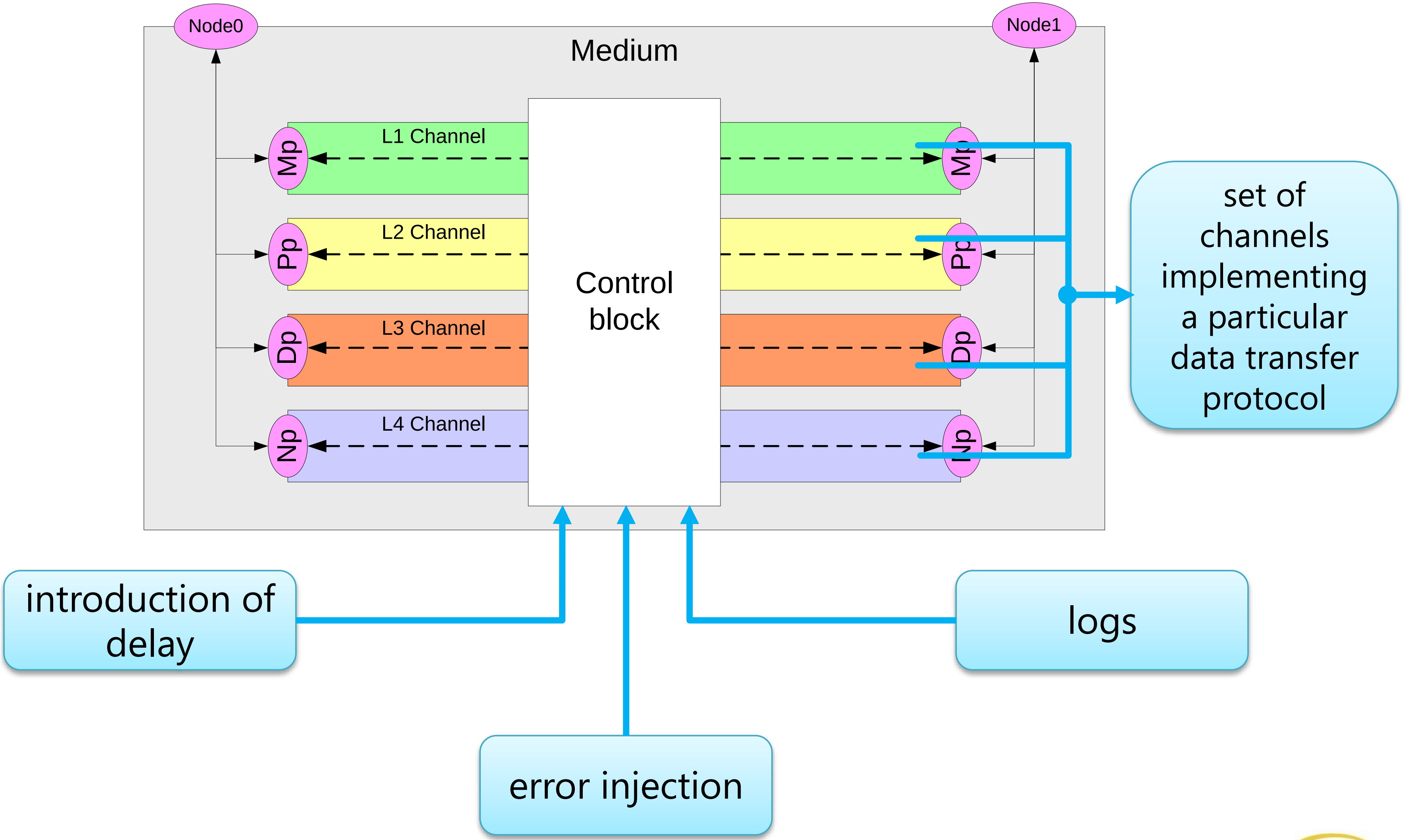
Modeling Core

- Each Intermediate Block can provide the following features:
  - managing data flows transmitted through the Communication Wrapper
  - parsing transmitted data – introduced for getting information about data exchange between two adjacent layers
  - making logs – used for monitoring of results
  - error injection – introduced for testing of error detection and/or error correction possibilities of a tested protocol
- Key issues for SDL model implementation:
  - each layer shall be represented by one SDL block
  - all required layers should be joined to one SDL system
  - each two adjacent layers of one node can be connected in two ways:
    - through an SDL channel only
    - through an appropriate Intermediate Block

Test Engine

- The Test Engine module is responsible for control of the SDL model simulation
- The Test Engine tasks:
  - Configuration of the SDL model Communication Wrapper before the start of test sequence
  - Configuration of the Medium. During this phase channel parameters such as the channel delay and the error injection are defined
  - Data exchange with the tested SDL model. The Test Engine operates in accordance with a protocol of the chosen layer and uses services of the layer below

The Medium



set of channels implementing a particular data transfer protocol

introduction of delay

logs

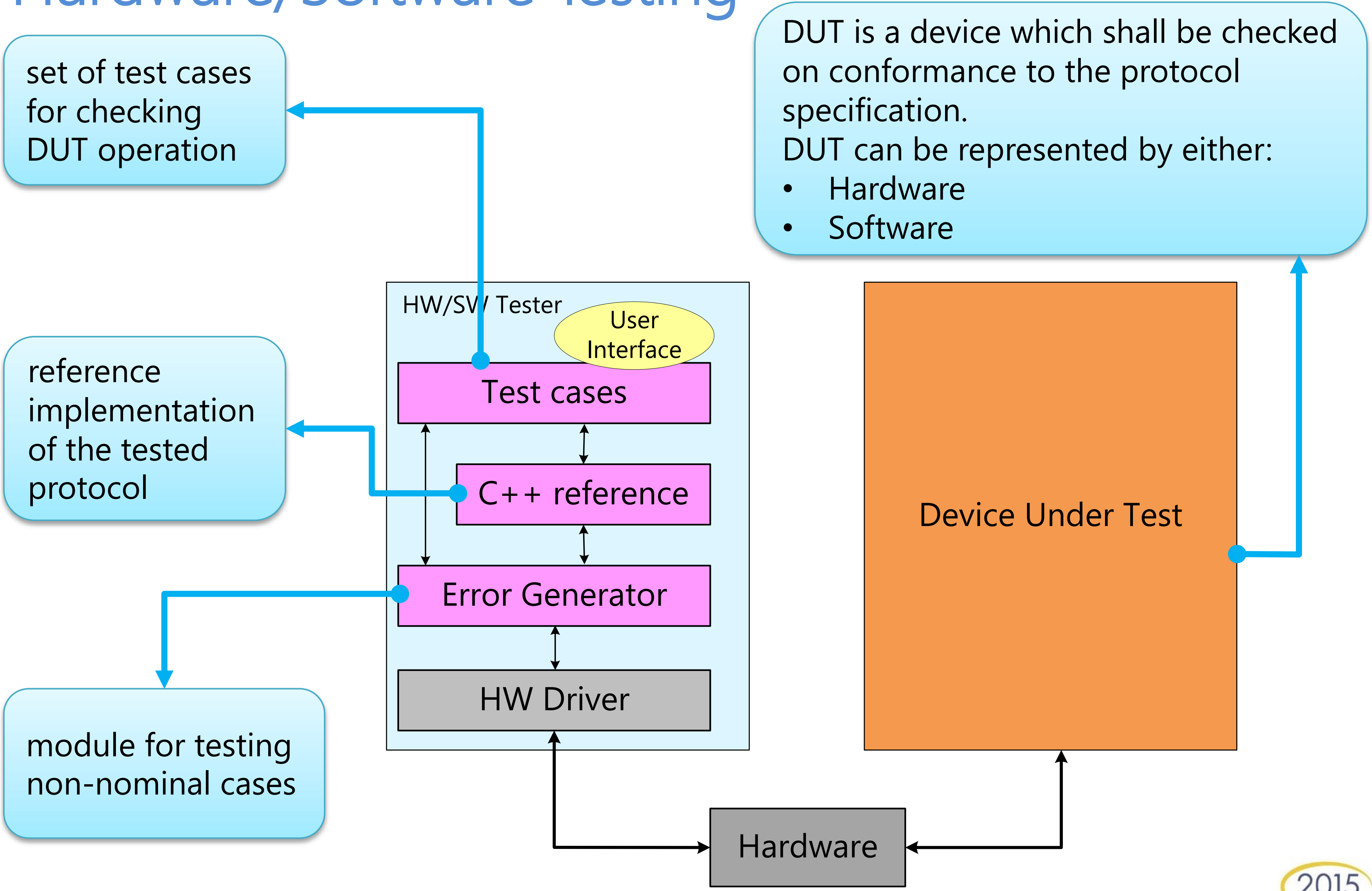
error injection

C++ Reference Model

Reference Code

- The reference code is a software implementation of a protocol in C++ language;
- **The objective of development:** create a reference for the programmers, who will implement the protocol in the software as well as use it as a part of a joint hardware/software tester;
- The C++ reference code describes:
  - the logical structure of the protocol,
  - its interfaces,
  - all internal mechanisms.
- This reference code can be used for:
  - studying of the protocol functionality;
  - translation into the other programming language;
  - implementation of a protocol in the software;
  - hardware/software testing.

Hardware/Software Testing


set of test cases for checking DUT operation
reference implementation of the tested protocol
module for testing non-nominal cases
HW/SW Tester
User Interface
Test cases
C++ reference
Error Generator
HW Driver
Hardware
Device Under Test

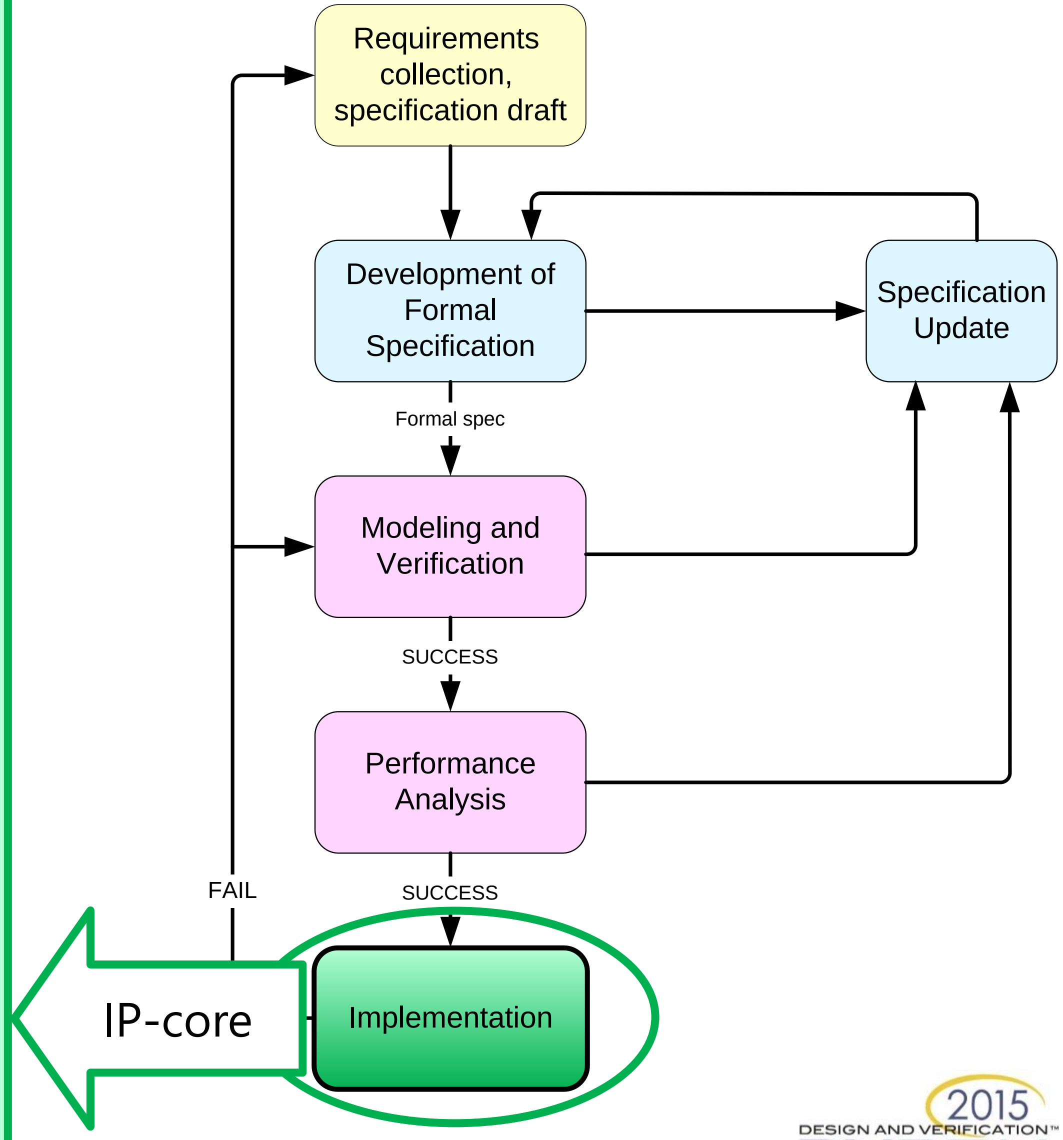
DUT is a device which shall be checked on conformance to the protocol specification.
DUT can be represented by either:

- Hardware
- Software

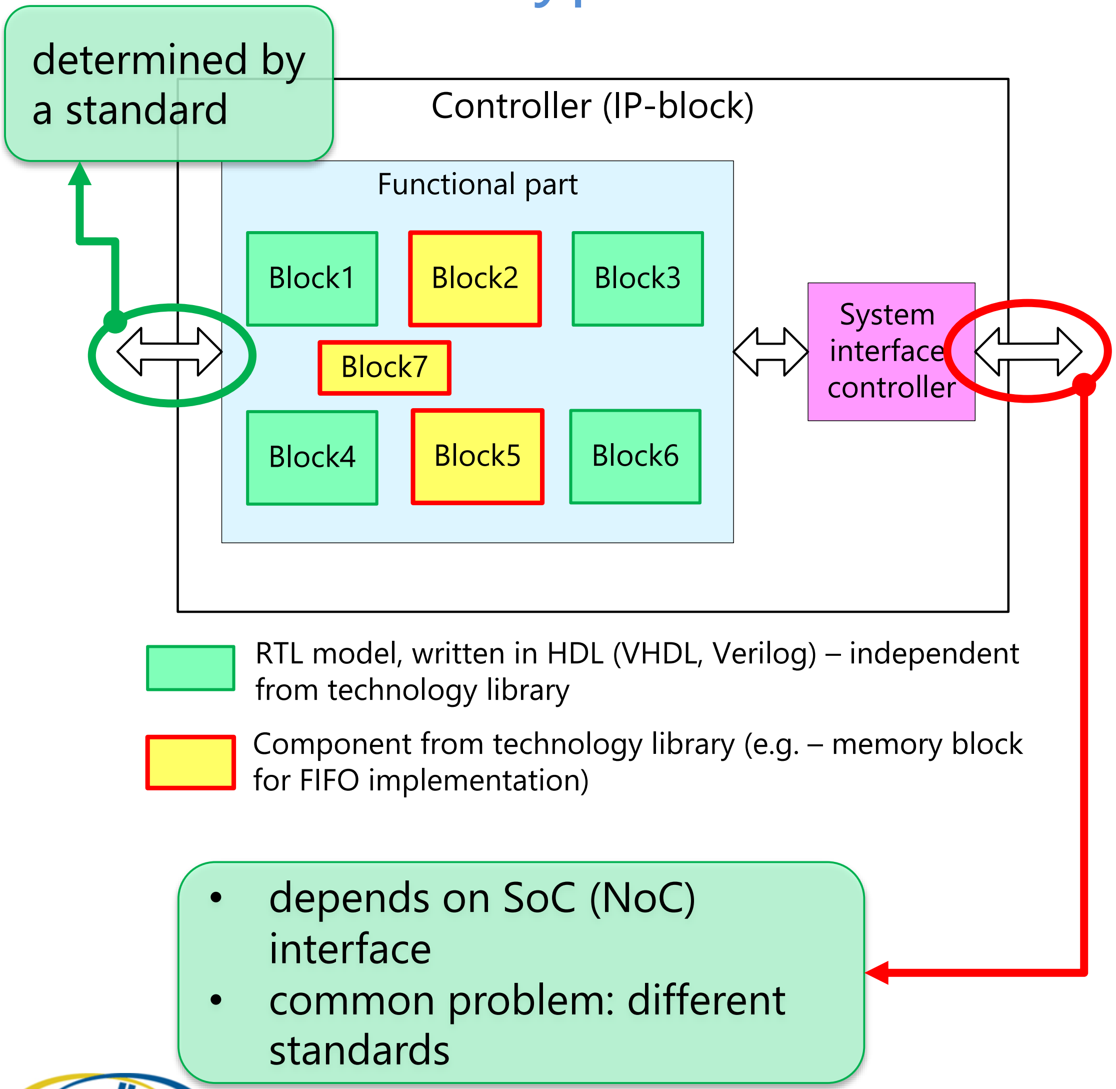
IP-Core Development

IP-core Development

- IP-core (or IP-block) is a reusable unit of logic, cell, or chip layout design
- **The objective of development**:
  - check the hardware implementation of the protocol,
  - check the operability of the protocol's mechanisms,
  - evaluate hardware costs.
- In our methodology we use **VHDL** for implementation of IP-blocks
- **Results of this stage:**
  - IP-block area estimation;
  - Clock frequency estimation;
  - Power consumption characteristics;
  - Protocol ready or not for the hardware implementation.


Requirements collection, specification draft
Development of Formal Specification
Specification Update
Formal spec
Modeling and Verification
SUCCESS
Performance Analysis
FAIL
SUCCESS
IP-core
Implementation

The protocol controller implementation: Typical structure of IP-block



determined by a standard

🟩 RTL model, written in HDL (VHDL, Verilog) – independent from technology library

🟨 Component from technology library (e.g. – memory block for FIFO implementation)

- depends on SoC (NoC) interface
- common problem: different standards

Main concerns on reusability in different network-on-chip (NoC) projects:

- Varied technologies (FPGA, ASIC), different technology process (libraries);
- Different standards of NoC system interface (e.g. AXI, OCP).

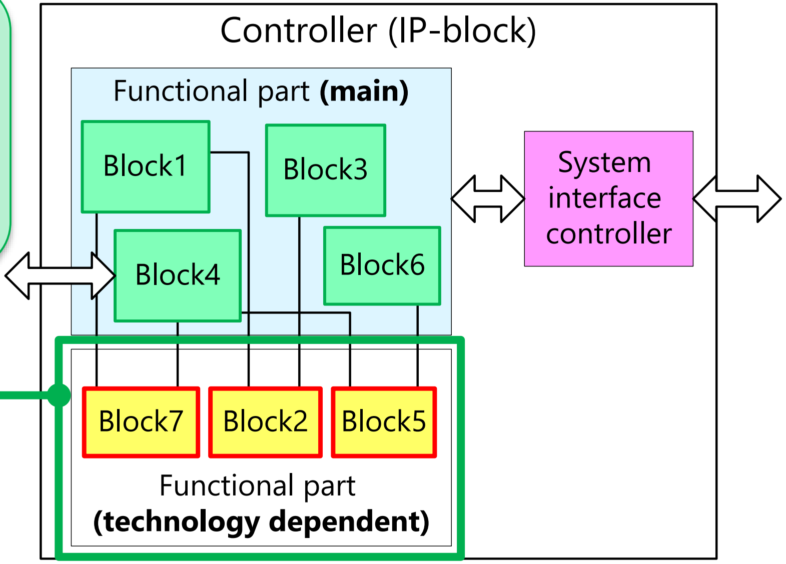
Issue of Varied Technologies

1

- Separation of technology dependent sub blocks to a particular “tech” block,
- Development the special “tech” block for used technologies

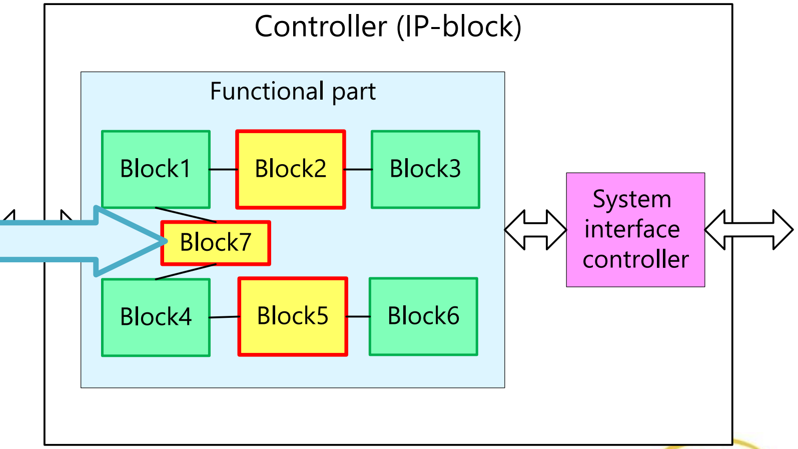
2

using “if… generate” for every technology dependent sub block


Controller (IP-block)
Functional part **(main)**
Block1
Block3
Block4
Block6
System interface controller
Block7
Block2
Block5
Functional part
(technology dependent)

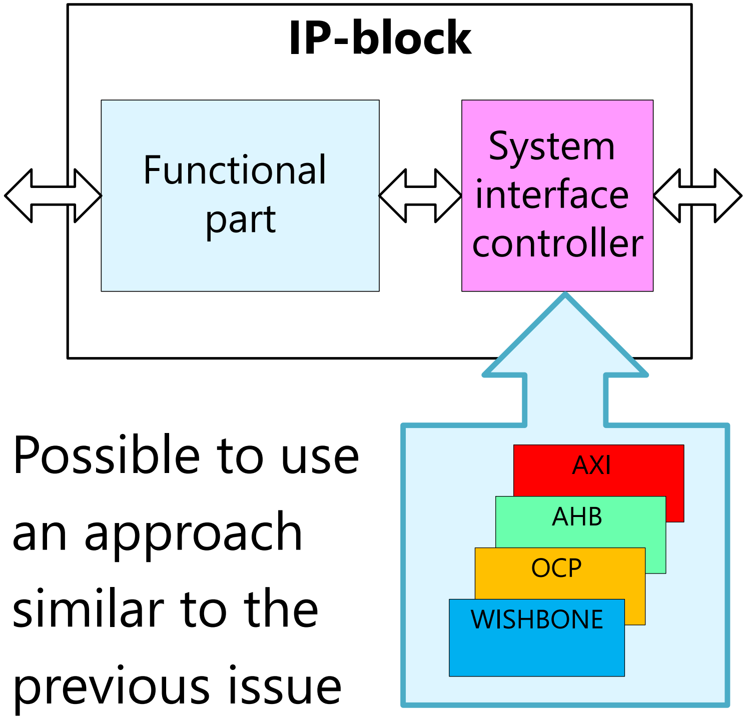
```
A_mem: for I in 1 to 1 generate
A_FPGA_Virtex_5: if tech_parameter = fpga_virtex_5 generate
A_buf: xRAMB16_S36_S36
port map(DOB=>QB_copy(31 downto 0),
          DOPB=>QB_copy(35 downto 32),
…
end generate

A_ASIC_1_90: if tech_parameter = ASIC_1_90 generate
A_buf: SRAM16x36
Port map(DB =>QB_copy,
 …
end generate;
end generate;
```



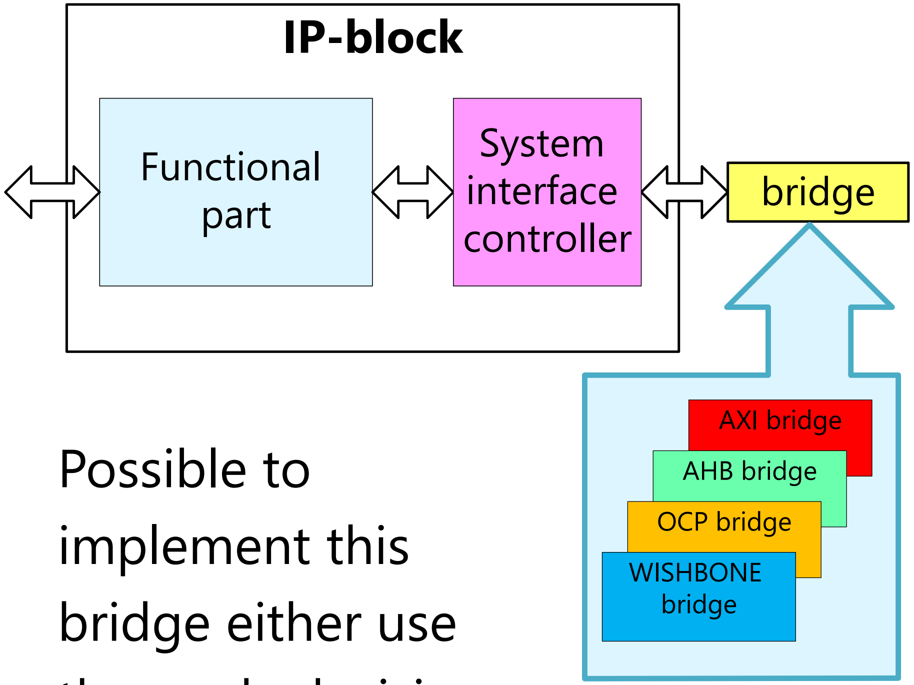
Issue of Different NoC System Interface Standards

1 Development of the **special block of a system interface controller** for every communication standard



Possible to use an approach similar to the previous issue

2 Utilize a **Bridge** between the IP-block and communication system



Possible to implement this bridge either use the ready decision from Design&Reuse

OUR EXPERIENCE

Modeling Directions in Projects

SDL

- UniPro (MIPI Alliance)
- SpaceWire (University of Dundee, ESA)
- SpaceWire-RT (FP7 project)
- STP-ISS protocol (JSC Information Satellite Systems)

SystemC

- UniPro, PIE (MIPI Alliance)
- SpaceWire, RMAP, STP (University of Dundee, ESA)
- SpaceWire-RT
- STP-ISS protocol, Plug-n-Play (JSC Information Satellite Systems)

SDL/SystemC

- UniPro
- SpaceWire-RT
- SpaceWire

C++ reference

- STP-ISS protocol

Thank you!
Questions?