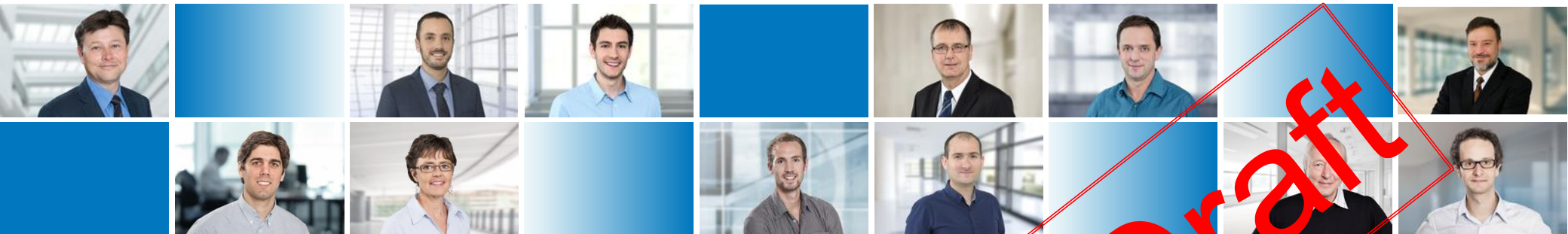


Modelling and Validation Framework for Efficient Development of Consumer Appliances for Mass Production

Dr. Michael Gortchacow, Dr. Julian Yeandel and Dr. Igor Kaitovic



Helbling Technik
Innovation, together we do it

Outline

- ❑ Introduction and Motivation
- ❑ Methodology Blueprint
- ❑ Case Study: DC Motor Controller
- ❑ Conclusions

Draft

Introduction: Challenges and Motivation

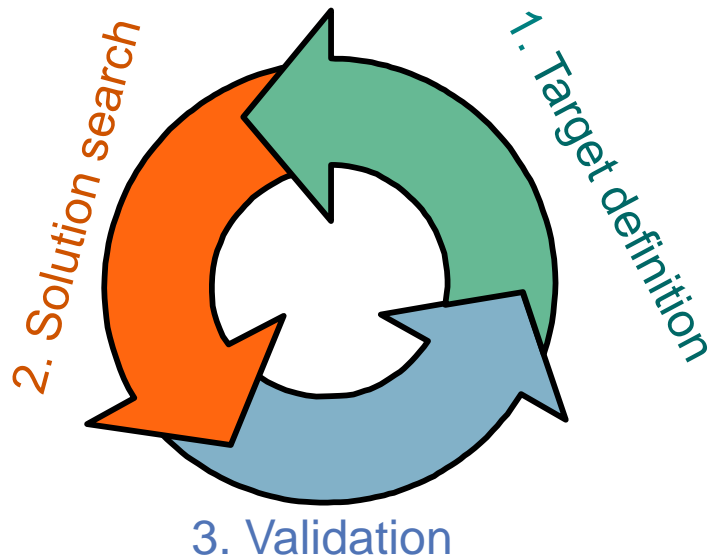
- Development of consumer appliances is increasing challenging due to:
 - Their growing complexity
 - Strong competition and high customer expectations
 - Low-cost requirements and tight schedules
 - Challenges related to the mass production

- As modern customer appliances are based embedded systems, ensuring high-quality of embedded software is of paramount importance

Draft

Introduction:

Embedded SW for Consumer Appliances



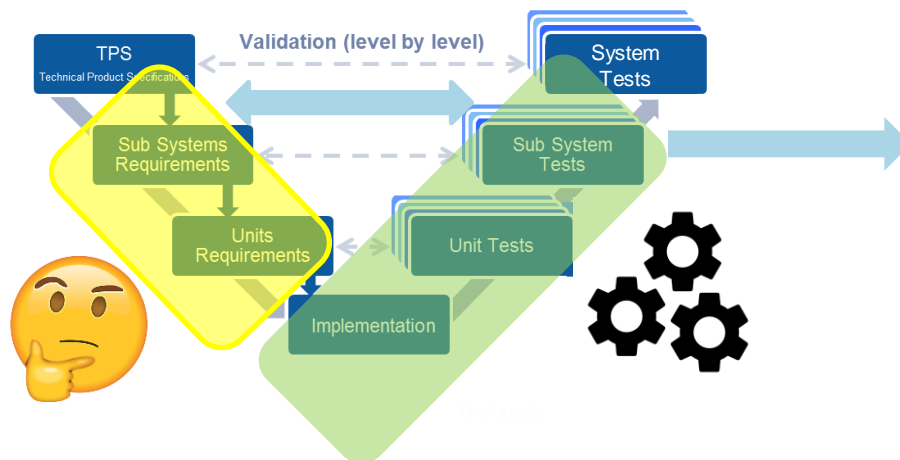
- ❑ Embedded SW development is an iterative process
- ❑ Validating the SW on a physical device after each iteration is:
 - Impractical
 - Time-consuming and
 - Sometimes impossible
- ❑ Modeling & simulation is a powerful tool to decrease the validation time and thus to significantly improve the overall efficiency

Draft

Introduction: Modeling & simulation

Model analysis:

- Simulation (experiment on a model) to understand the system or answer specific questions
- Sensitivity Analysis
- Model-Based Diagnosis
- Model Verification and Validation:
 - Are we building the system right?
 - Are we building the right system (requirements)?



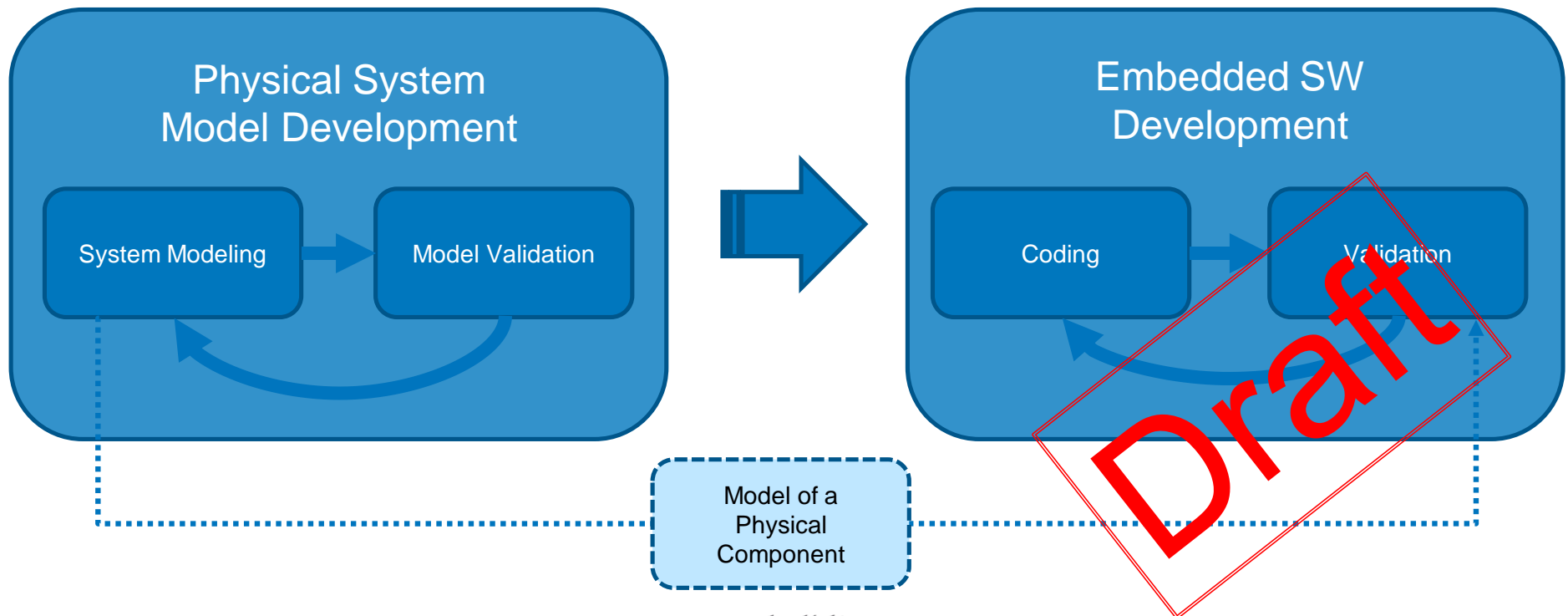
This gives inputs for:

- **Risk analysis at an early stage** → modifications in the system architecture are less costly at early stages.
- Pinpoints weak and strong points of system early on
- Optimization of system
- Qualitative selection of critical tests to be performed on eventual prototype
- **Deep understanding of behavior of the system & reduced likelihood of unexpected behavior/failure**

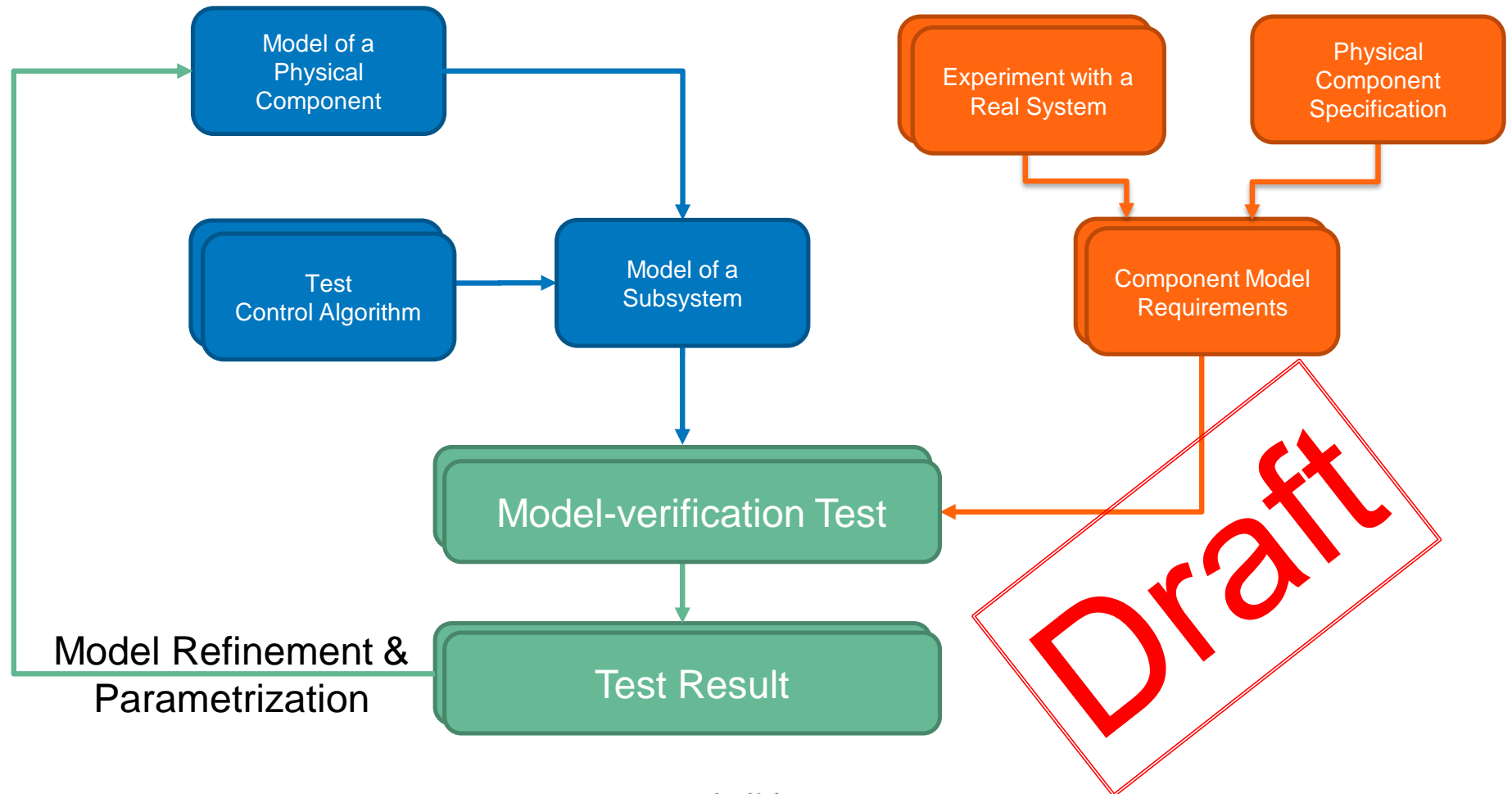
Draft

Methodology: Overview

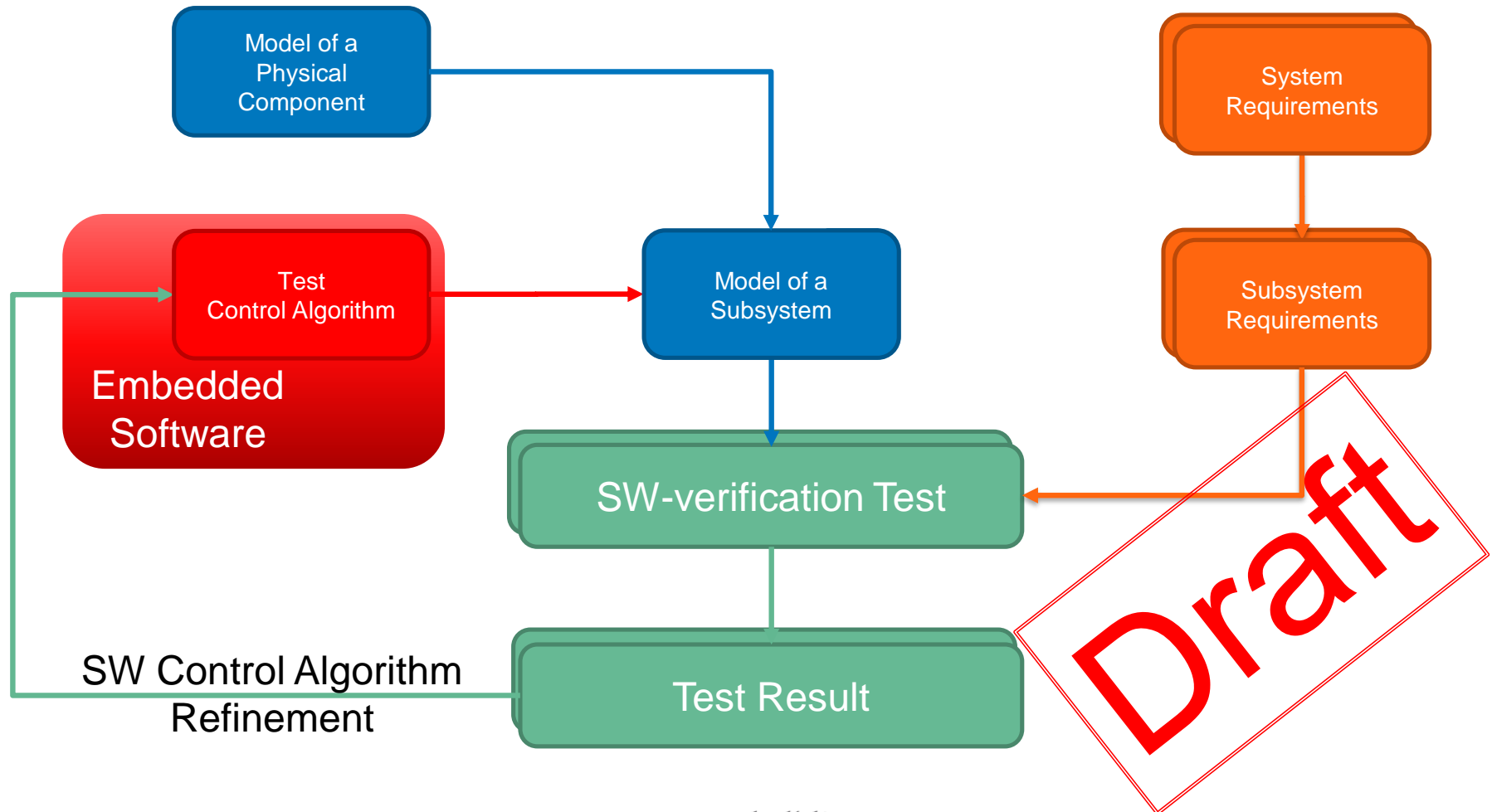
- Modeling of a physical system and its components
- SW development and verification using the model
- Automated SW testing after each SW change



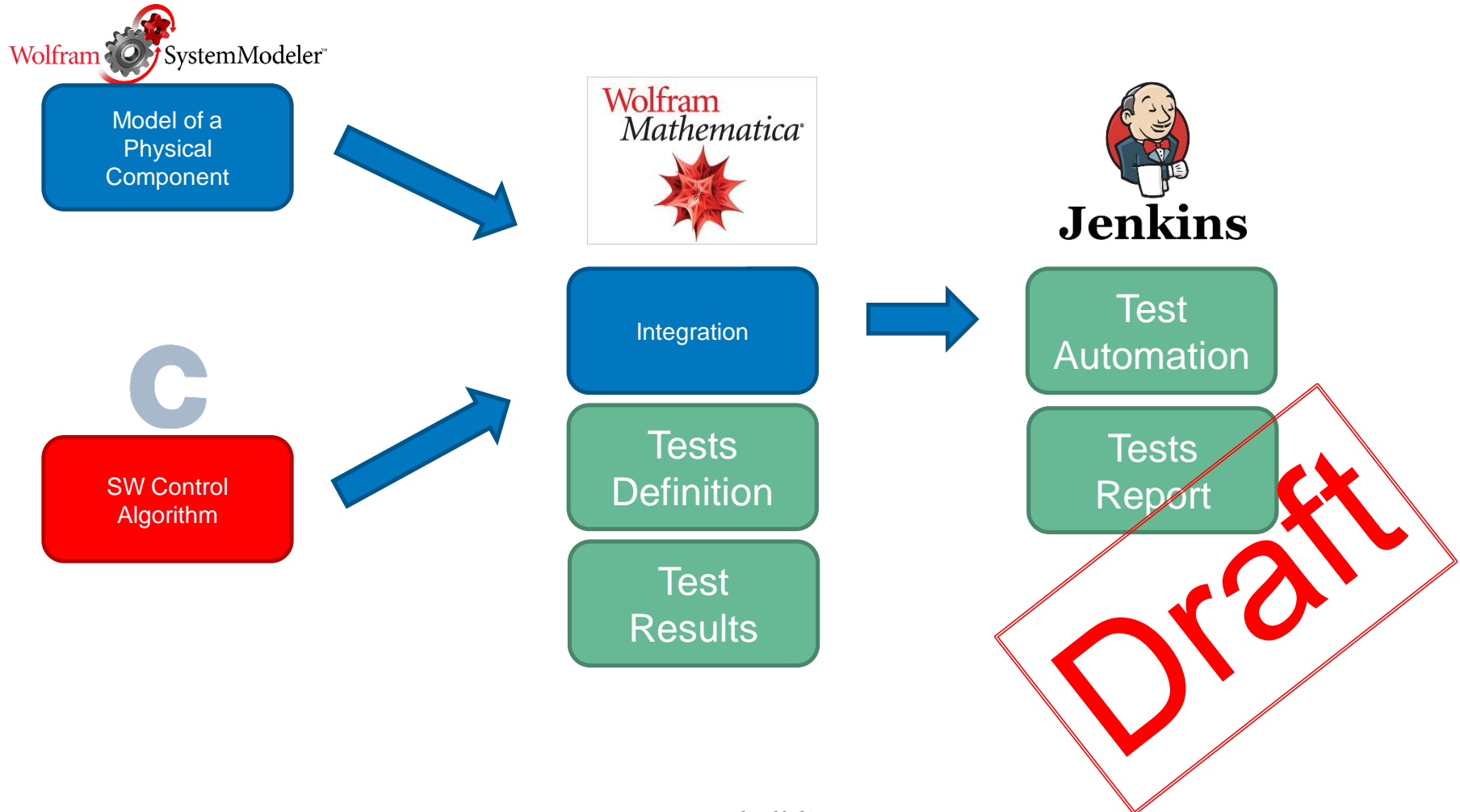
Methodology: Physical System Modeling



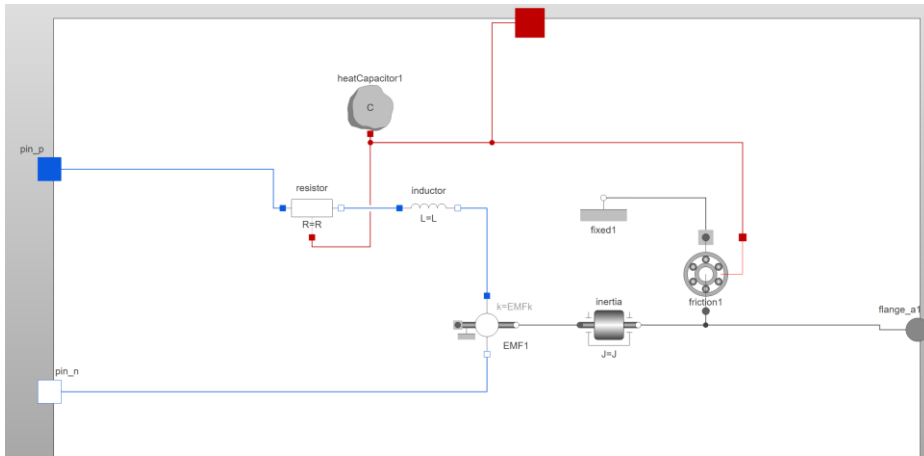
Methodology: SW Development and Validation



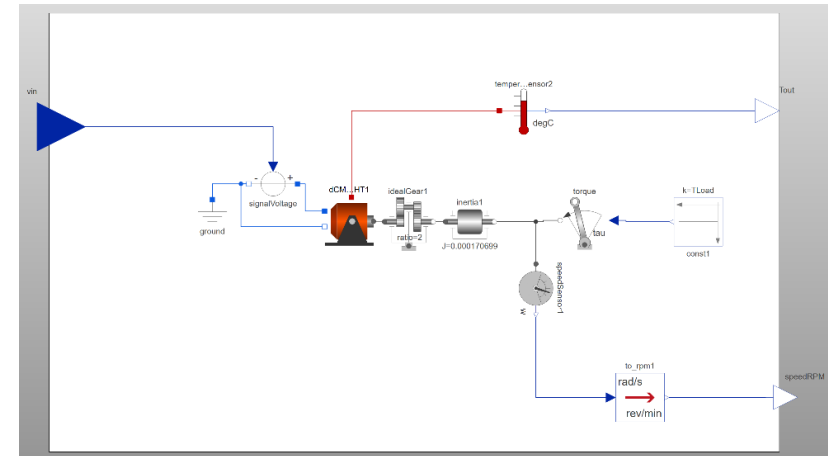
Methodology: Tool Chain



Case Study: DC Motor Controller



DC Motor Model



Subsystem model

Draft

Case Study: DC Motor Controller

- testsCompiledByModelica4
 - BaseController
 - PIcontinuousController
 - PIDdiscreteController
 - PldiscreteController
 - PIDcontinuousController
- ExternalPID4
- MotorControllerExternal4
 - MotorCharacteristicCurve

SW Controllers and Tests

```

include ExternalPID4 "pid controller using external c code and compiled with symbolexporter for using General Data Structure"
model ExternalController
  external BasicCharacteristic = 0.01, K = 0.05, Td = 0, Tl = 0.1;
equation
  when ramp1(t, Ts) then
    output = Modelica.ExternalFunctions.PIDdiscrete(ref, rBuSystem_Discrete_Ext, K, Kd, Td, Tl);
  and when
    and when
  end when
end ExternalController;

package Modelica
  "Package containing external C functions"
  package ExternalFunctions "Package containing external C functions"
  import Function FPD "Function for PID controller";
  input Real error "Error signal for PID controller";
  input Real K "Gain";
  input Real Kd "Derivative gain";
  input Real Td "Integral time constant";
  input Real Tl "Load time constant";
  output Real signal "PID output";
end ExternalFunctions;

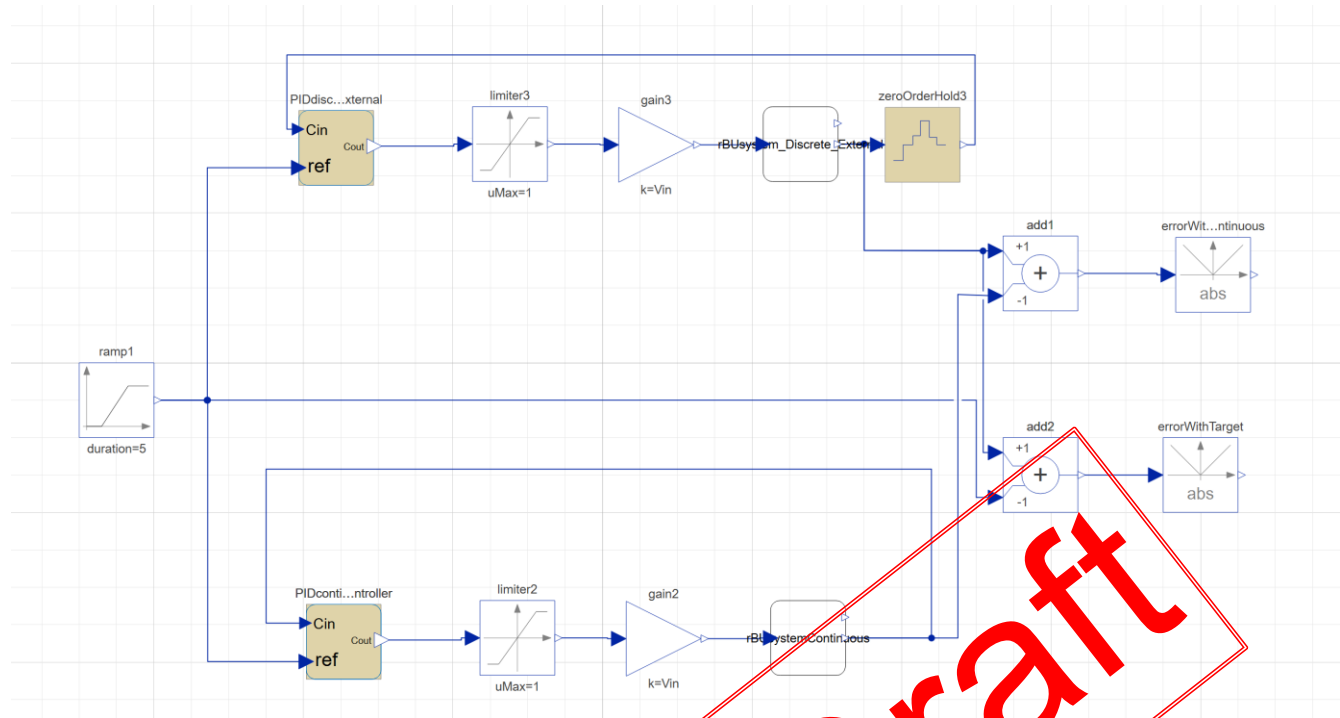
external "c" signal = external_controller(error, Ts, K, Td, Tl) annotation(Include = "ExternalController.lib", Library = "ExternalPID4");

extend Modelica.Blocks.Package;
end ExternalFunctions;

model PIDdiscreteController
  Cin: Real;
  Cout: Real;
  ref: Real;
  limiter2: Real;
  gain2: Real;
  rBuSystemContinuous: Real;
  limiter3: Real;
  gain3: Real;
  zeroOrderHold3: Real;
  add1: Real;
  abs1: Real;
  errorWithTarget: Real;
  add2: Real;
  abs2: Real;
end PIDdiscreteController;

model PIDcontinuousController
  Cin: Real;
  Cout: Real;
  ref: Real;
  limiter2: Real;
  gain2: Real;
  rBuSystemContinuous: Real;
  limiter3: Real;
  gain3: Real;
  zeroOrderHold3: Real;
  add1: Real;
  abs1: Real;
  errorWithTarget: Real;
  add2: Real;
  abs2: Real;
end PIDcontinuousController;

```



Subsystem simulation

System integration script

Case Study: DC Motor Controller

```

model
sin = SystemModelSimulate[tests.MotorCharacteristicCurve];
speed = sin["mechanicalPowerSensor.relSpeedSensor.u_rel"][[1]];
efficiency = sin["efficiency.y"][[1]];
torque = sin["ramp.y"][[1]];
current = sin["DCMotor_H11.resistor.p.i"][[1]];

data = sin["RawData", {"ramp.y", "mechanicalPowerSensor.relSpeedSensor.u_rel", "DCMotor_H11.resistor.p.i", "efficiency.y"}][All, 1, 2];

workingPoint = (Mean / # (Select[data, Round[#[[1], 0.0001] == 0.11 &]])) {1000,  $\frac{60}{2\pi}$ , 1, 100};
stall = (Mean / # (Select[data, Round[#[[2], 1] == 0 && #[[1] > Evaluate@ (workingPoint[[1] / 1000 &]])])) {1000,  $\frac{60}{2\pi}$ , 1, 100};
noLoad = (Mean / # (data[All, Evaluate@Latten@Position[data[[2]], Max[data[[2]]]]]) {1000,  $\frac{60}{2\pi}$ , 1, 100};

Evaluate@ {workingPoint, stall, noLoad}

ExportedOutput
{{110, 453.024, 0}, {8032, 0, 10670}, {5.564, 22.313, 0.225}, {09.28, 0, 0}}

ExportedMessages
{}

TestOptions
[TestID = "motor model", SameTest = (Norm[#[[1] - #[[1]]] < 30 && Norm[#[[2] - #[[2]]] < 250 && Norm[#[[3] - #[[3]]] < 0.5 && Norm[#[[4] - #[[4]]] < 0.05 &)]

Success
Add Messages Add Options Run

```

```

model
(* compile source code found within the class *)
dir = FileNameDropSystemModel["DLESystemModeling.Components.PIDs.ExternalPID4.Utilities.ExternalFunctions"]["SourceFile"];
RunProcess[dir <- "make.bat", "StandardOutput", ProcessDirectory = dir];

(* compile and run model code *)
sin2 = SystemModelSimulate[tests.MotorControllerExternal4];
errorTarget = sin2["RawData", {"errorsWithTarget.y"}];
Table[Max[Select[errorTarget[[1], #[[1] > tmin &]][All, 2]], {tmin, 0.1, 2}]

ExportedOutput
{100, 30}

ExportedMessages
{}

TestOptions
[TestID = "embedded code controller, ramp up overshoot", SameTest = ( #[[1] < #[[1]] && #[[2] < #[[2]] &)]

Success
Add Messages Add Options Run

```

Tests definition

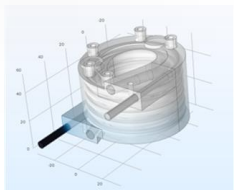
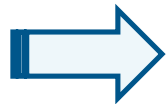
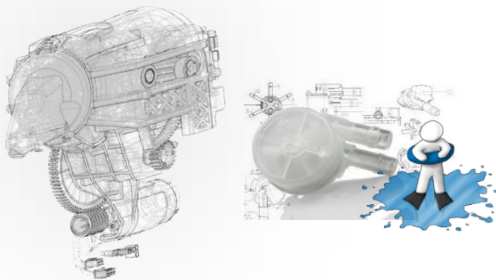
Test Automation

Jenkins Test Results for 'wit' showing 7 tests passed. A large red 'Draft' watermark is overlaid on the image.

Test name	Duration	Status
DLESystemModeling Library loaded	16 sec	Passed
Library loaded	0.29 sec	Passed
Load SystemModeler	37 sec	Passed
Modelica Library loaded	0 ms	Passed
compiler	15 ms	Passed
embedded code controller_ramp up overshoot	26 sec	Passed
motor model	41 sec	Passed

Conclusions

- ❑ Development of consumer appliances requires systematic application of Model Based Systems Engineering (MBSE) principles through all the stages of the product development
- ❑ Using our approach to embedded SW development and validation, we can significantly improve our overall efficiency and thus meet the challenge of balancing the time, quality and cost
- ❑ Successful application of this approach to a wide range of products has resulted in high-quality consumer appliances and customer satisfaction



Thank You for the Attention

- Dr. Michael Gortchacow
Principal Engineer
Helbling Technik Bern (HTKB)
Home & Office Appliances (HOA)
michael.gortchacow@helbling.ch
- Dr. Julian Yeandel
Head of Development
Smart Home, Business & Lifestyle Appliances
at HTKB HOA
julian.yeandel@helbling.ch
- Dr. Igor Kaitovic
Embedded Software Engineer
at HTKB HOA
igor.kaitovic@helbling.ch

Helbling Technik Bern AG

Stationsstrasse 12
CH-3097 Liebefeld
www.helbling.ch

