# MODELLING AN INJECTION MOULDING MACHINE

Using the Vienna Development Method

***Till Boettjer***, *Michael Sandberg, Peter G. Larsen, Hugo D. Macedo*

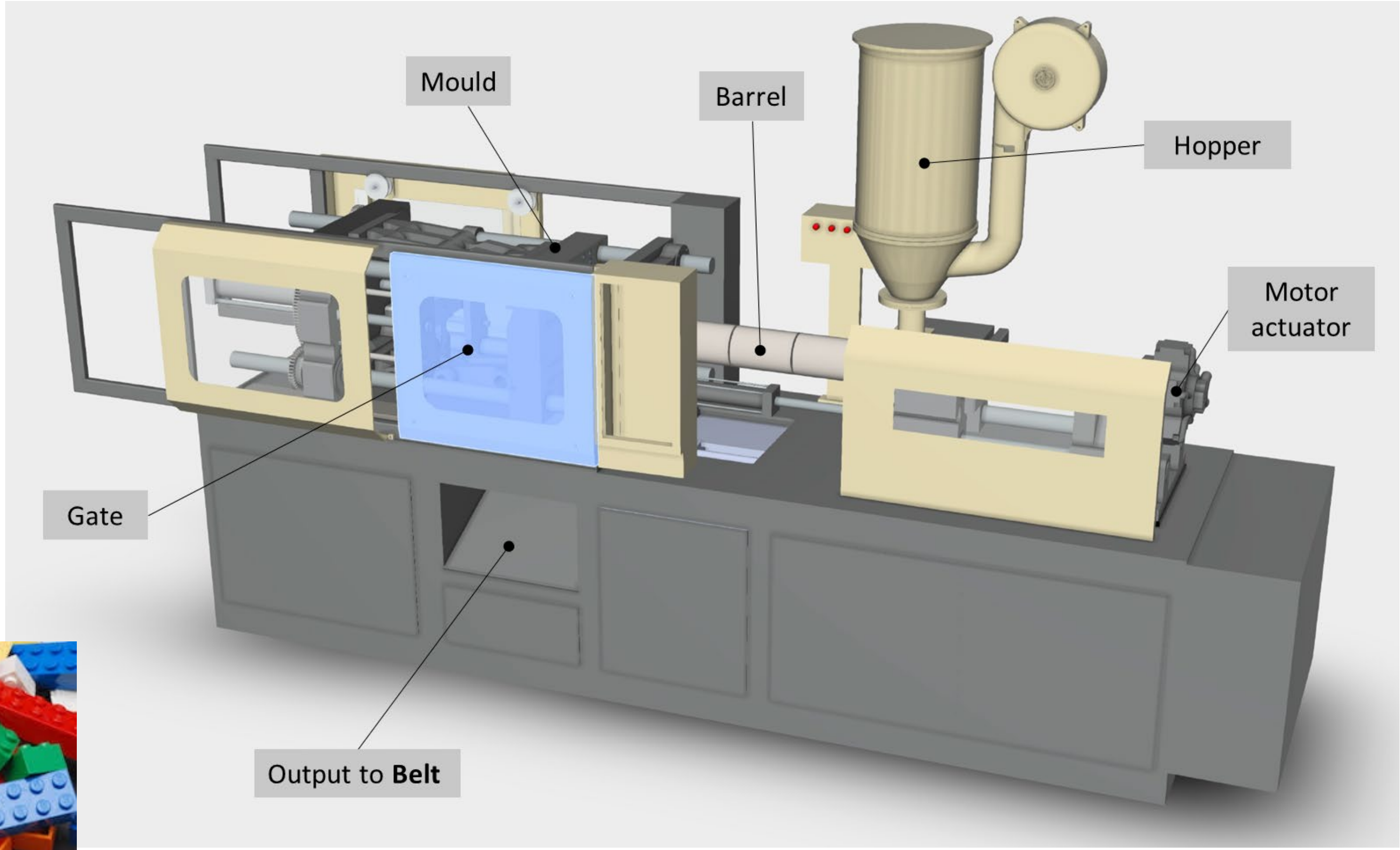
# AGENDA

- ❑ Motivation
- ❑ Injection Moulding Machine (IMM) & process
- ❑ Continuous IMM model
- ❑ VDM++ model of an IMM
- ❑ Conclusion

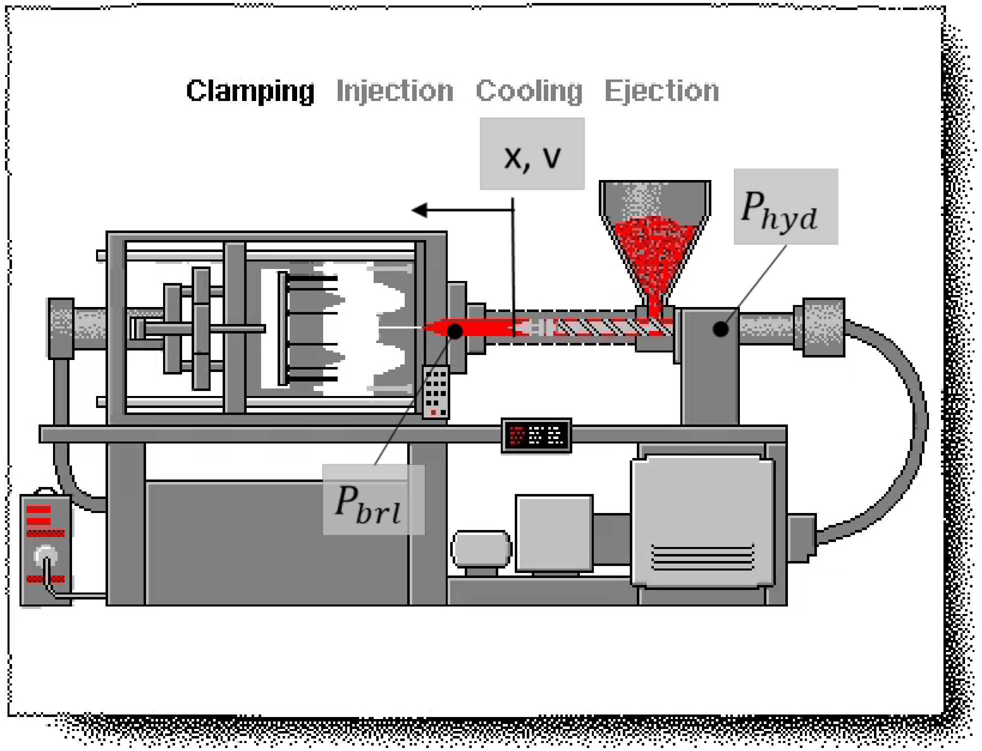
# INJECTION MOULDING MACHINE

PhD, Digital Twins in manufacturing using Co-simulation

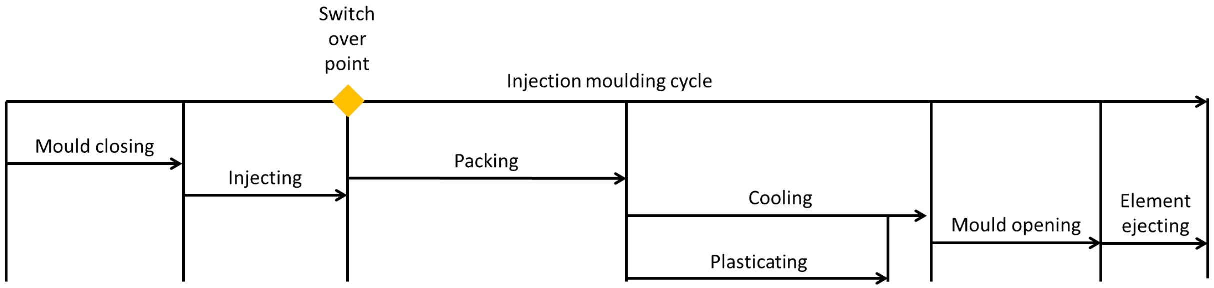
Collaboration with one of the largest injection moulding companies in the world




# INJECTION MOULDING PROCESS




Source: https://www.docsity.com [Access: 16.10.2020]




- Plasticating granulate
- Filling cavity
- Packing and holding
- Cooling
- Releasing of molded part

# MOTIVATION

- IMM are highly computerized and hold a large opportunity for employing advanced control schemes
- Take a first step towards a Digital Twin for IMM using Co-simulation
- Model the behavior of an controller of an IMM and use VDM to check properties

**Property 1** - The machine must be idle when the gate is open.

**Property 2** - The screw must reach EndPositionFill. For the scenario, End-PositionFill equals 20.3 mm.

**Property 3** - The hydraulic pressure peak must occur at PeakTimeIntervall and be Peak-PressureBand. For the scenario, these equal to 0.9< t <1.1s and 9±1MPa.

# CONTINUOUS TIME MODEL

Adapted the work from [1] - First principle model linking controllable parameters to measurable process variables

- Set of differential equations using Newton's Second Law and thermodynamic relationships
- Solved for hydraulic pressure, nozzle pressure, ram velocity, polymer flow and cavity pressure
- Used to extract three scenarios as input to the VDM++ model

[1] Woll, S.L., Cooper, D.J.: A dynamic injection-molding process model for simulating mold cavity pressure patterns. Polymer—Plastics Technology and Engineering36(5), 809–840(1997)

# OUTPUTS FROM CT MODEL

First order dynamic eq. of screw velocity (Filling)

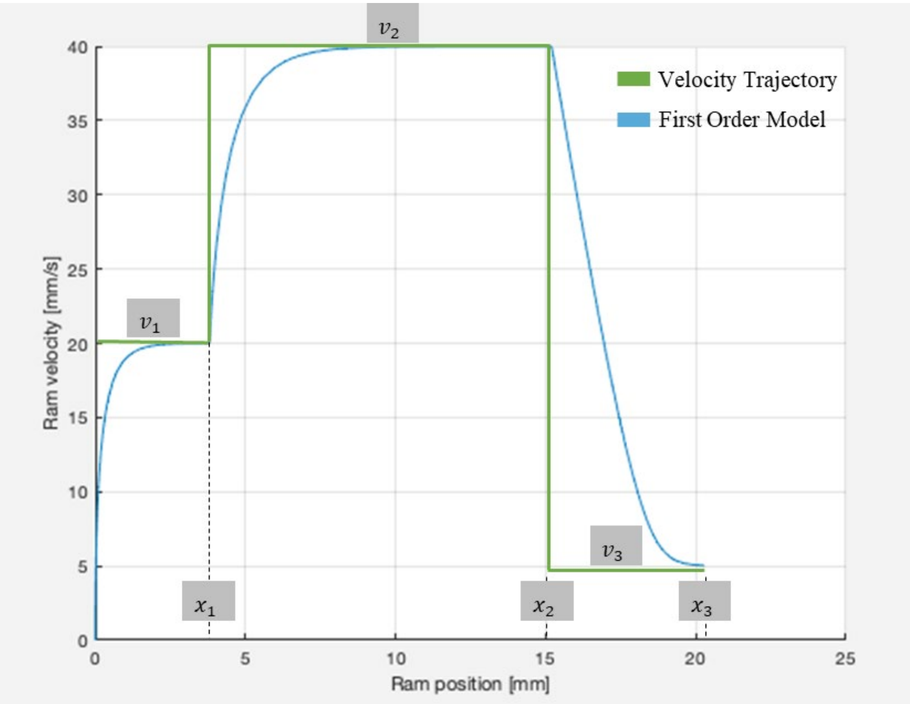
$$\tau_1 \left(\frac{dv}{dt}\right) + v = K_1 v_f$$

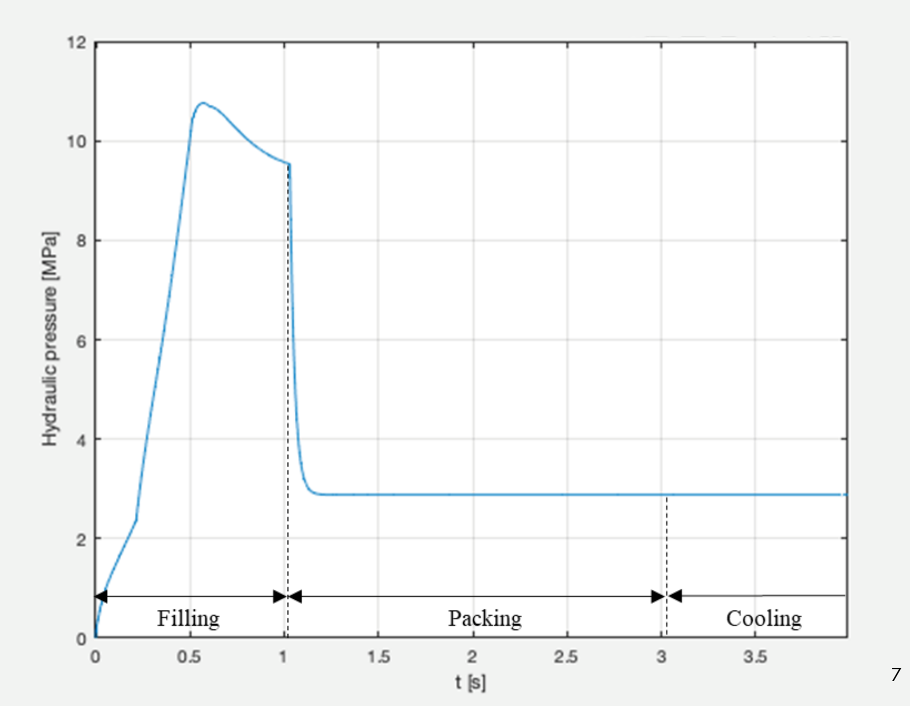
Newton's Second Law for finding hydraulic pressure

$$\left(\frac{dv}{dt}\right) = \frac{\sum F}{M} = \frac{P_{hyd}A_{hyd} - P_{brl}A_{brl} - C_f sign(v) - B_f v}{M}$$

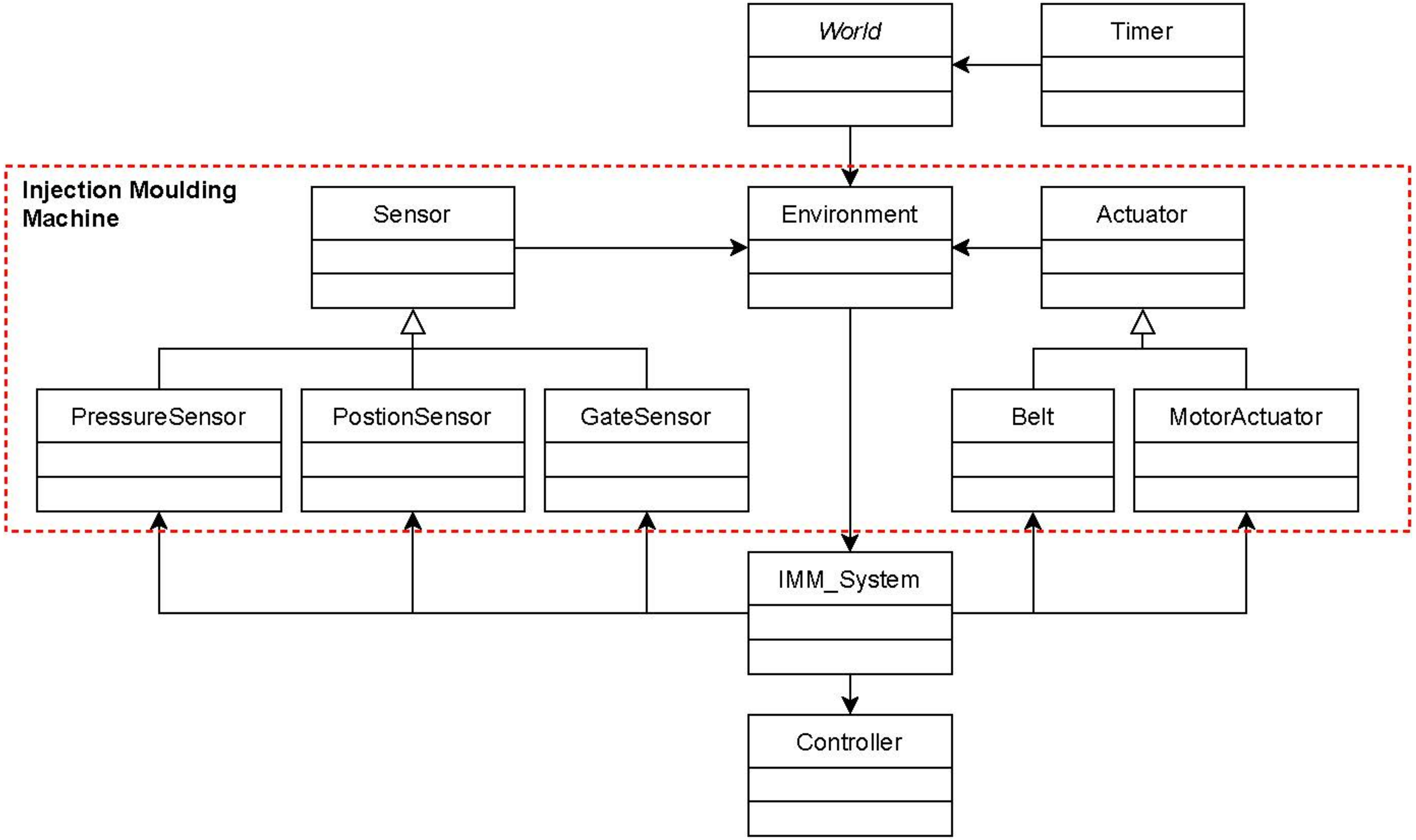
First order dynamics, for modelling transition to constant holding pressure (packing)

$$\tau_2 \left(\frac{dP_{hyd}}{dt}\right) + P_{hyd} = K_2 P_{hold}$$



# UML MODEL OF VDM++ MODEL

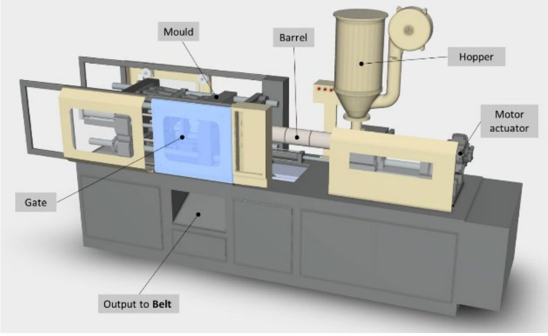


# EXAMPLE OF SAFETY ANALYSIS

```
types
  StateType    = <Filling> | <Packing> | <Cooling> | <Ejecting> |
       <Idle>;
  LockState    = <Open> | <Close>;
```

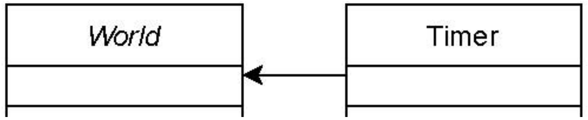
**Listing 1.1.** Highlights from the Controller class

```
instance variables

  ctl_state : StateType;
  lock : LockState;
    inv InterlockInvariant(lock,ctl_state)

functions

  InterlockInvariant : LockState * StateType -> bool
  InterlockInvariant (ls,s) == (ls = <Open>) => (s = <Idle>)
```



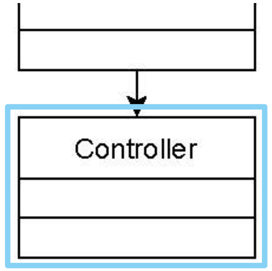

**Listing 1.2.** Instance variables used to assert safety

# EXAMPLE OF PROCESS CONTROL




```
public Step: () ==> ()
  Step() == (
   cases ctl_state:
      <Idle> -> return,
      <Filling>  -> FillStep(),
      <Packing>  -> PackStep(),
      <Cooling>  -> CoolStep(),
      <Ejecting> -> EjectStep() end;);
```

**Listing 1.3.** Different IMM operation stages



# EXAMPLE OF PROCESS CONTROL

```
public FillStep : () ==> ()
FillStep() == (
  -- Fetch sensor reading
  ctl_x := IMM_System`ps.getReading();
  -- Run control flow
  if (ctl_x < x1)
  then (IMM_System`imm.v_t := v1)
  elseif (ctl_x < x2)
  then (IMM_System`imm.v_t := v2)
  elseif (ctl_x < x3)
  then (IMM_System`imm.v_t := v3)
  -- Transition to Packing
  else Delta() )
  pre ctl_state = <Filling>;
```

**Listing 1.4.** Screw velocity setpoint assignment based on current position

# EXAMPLE OF QUALITY CONTROL

```
public EjectStep : () ==> ()
EjectStep() == (
  let success : bool = PartIsOK()
  in if (success) then
    ctl_bin := <OK>
     else
    ctl_bin := <Scrap>;
  pre ctl_state = <Ejecting>;
```

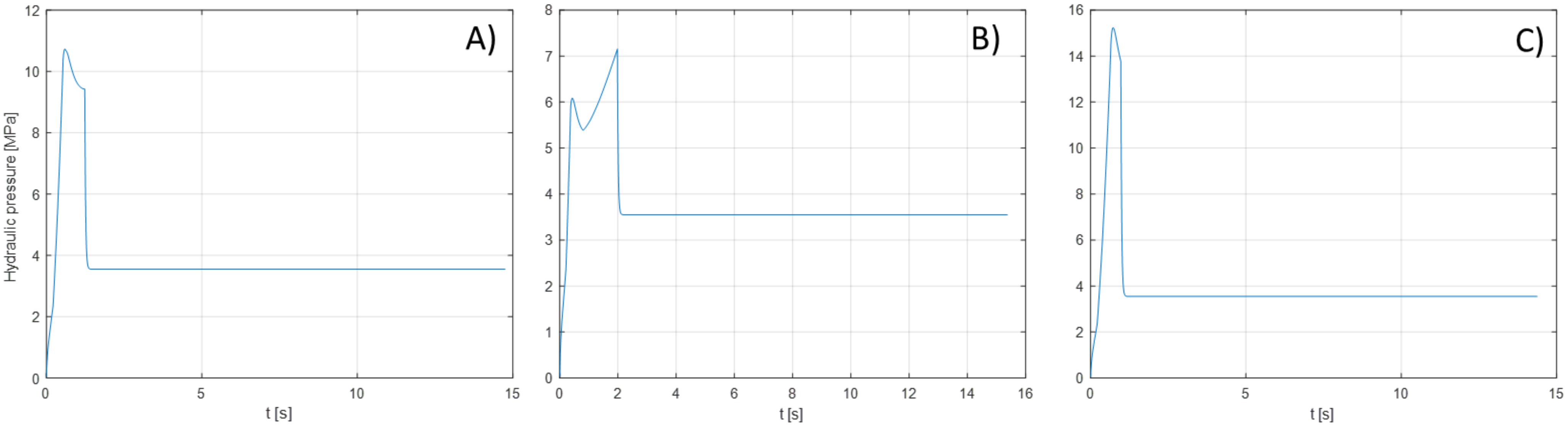
**Listing 1.6.** Return to initial state of the screw using PD control

```
pure PartIsOK : () ==> bool
PartIsOK () == return (ctl_x_max >= 20.3) and hyd_peak_ok and
     (hyd_peak_max >= 9.1);
```

**Listing 1.7.** Part of the quality control logic defined in the VDM specification

# VALIDATION OF MAIN FUNCTIONALITY

- ✓ Invariant and properties are satisfied
- ✓ Switching between injection phases
- ✓ Setting target velocities during filling
- ✓ Returning screw in cooling phase
- ✓ Auto-scrap of parts with quality issues



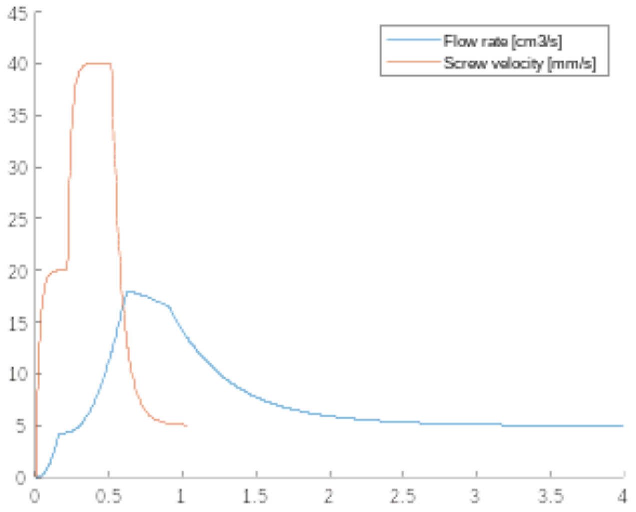
# CONCLUSION

- ☑ High level description of the control architecture of an IMM – including process, quality and safety control
- ☑ Preliminary validation of the VDM model based on 3 selected scenarios generated from a CT model created in Matlab
- ☑ First step towards a Co-simulation model of an IMM in a Digital Twin setting
- ❑ Enriching, refining and adding tests to the model
- ❑ Coupling of the models within a Co-simulation environment

# FUTURE: CO-SIMULATION OF IMM

**Contract**

## Continuous Time Model




**Monitored Variables**

Hydraulic pressure →

Ram Position →

**Controlled Variables**

← Target Ram Velocity

← Phase Change

← Auto Scrap

← Plastification speed

## Discrete Event Model

```
class InjectionMachine
types

  stateType = <Filling> | <Packing> | <Cooling> | <Ejecting> | <Idle>;
  LockState  = <Open> | <Closed> ;

values

instance variables

  public env_x : real; -- position of screw
  public env_v : real; -- velocity of screw
  public env_p_hyd : real;      -- hydraulic pressure
  public v_t : real; -- target velocity during filling
  public env_p_hyd_t : real; -- target pressure == holding pressure

  public env_state : stateType; -- current phase
  env_lock   : LockState;

operations

  public InjectionMachine : (real) ==> InjectionMachine
  InjectionMachine(x0) == (
    env_state := <Idle>;
    env_lock      := <Closed>;
    env_x := x0;
    env_v := 0;
    env_p_hyd := 0;
    v_t := 0; -- [mm/s]
    env_p_hyd_t := 0;
  );
```