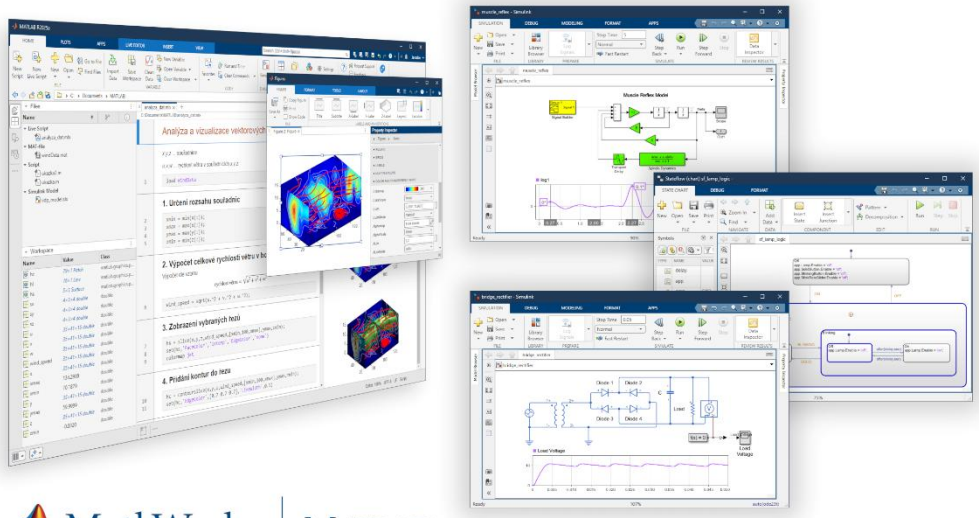
HUMUSOFT

MATLAB a Simulink Master Class: Vývoj řídicích systémů s využitím modelů



MathWorks Authorized Reseller | MATLAB & SIMULINK

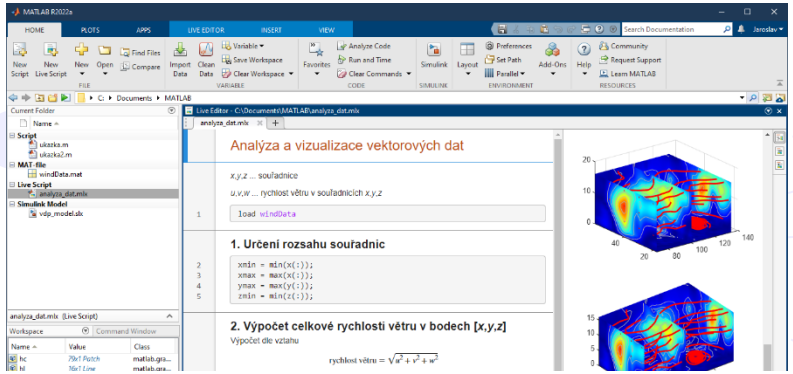
Jaroslav Jirkovský

jirkovsky@humusoft.cz

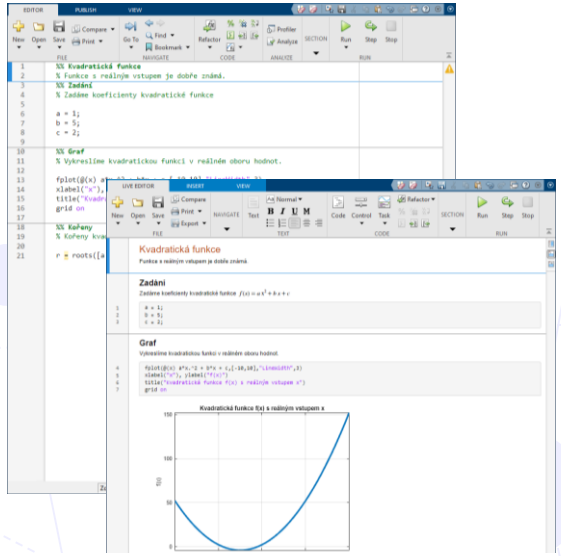
www.humusoft.cz
info@humusoft.cz

www.mathworks.com

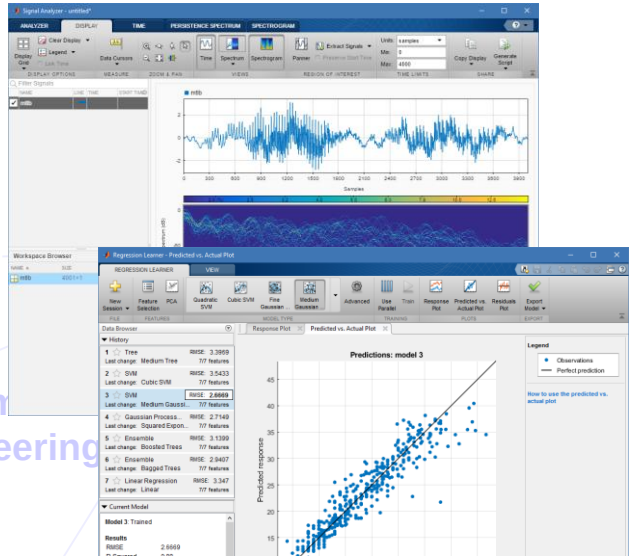
Co je MATLAB



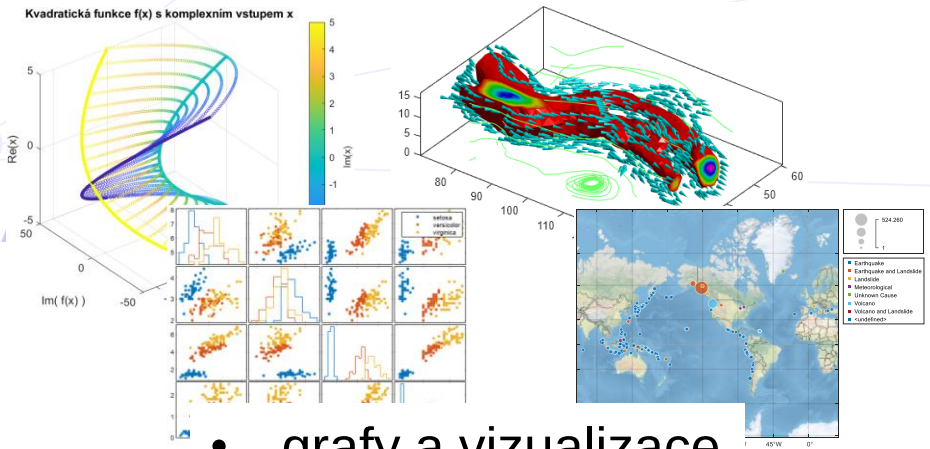
- inženýrský nástroj
- interaktivní prostředí
- vědecké a technické výpočty



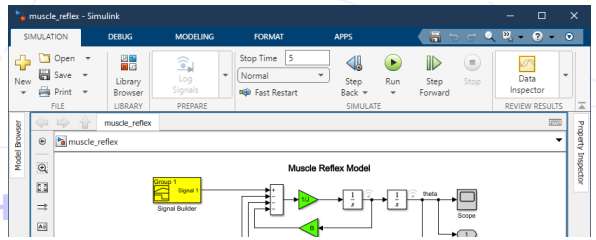
- výpočetní nástroje
- otevřený systém



- grafické aplikace (Apps)



- grafy a vizualizace



- modelování systémů
- simulace a analýza
- Model-Based Design

- 100+ aplikačních knihoven
- 10 000+ výpočetních funkcí
- jednotná dokumentace

- propojení s externím hw / sw
- vývoj samostatných aplikací

Kde se MATLAB používá ...

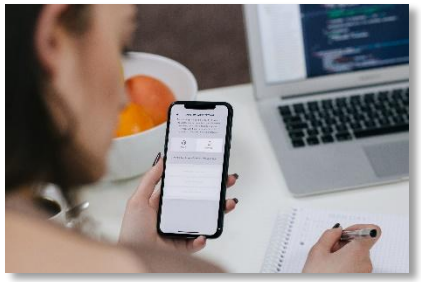
Letecký průmysl

Automobilový průmysl

Přírodní vědy

Biotechnologie a farmacie

Komunikace



Elektronika

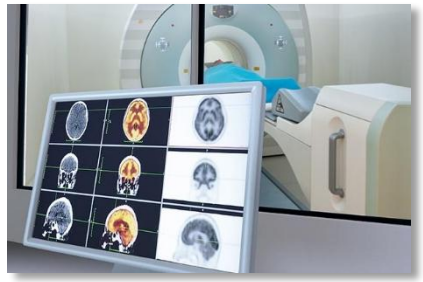
Energetika

Finanční služby

Výrobní stroje

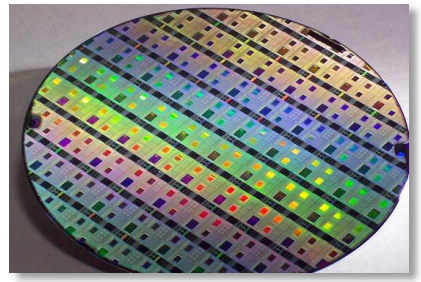
Medicína

Těžební průmysl



Neurovědy

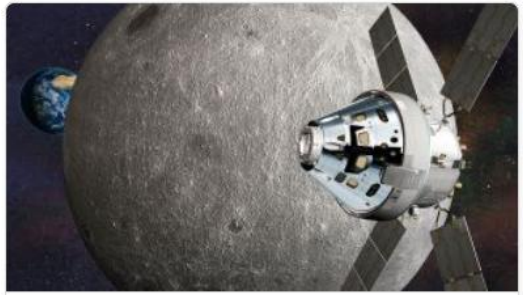
Železnice a doprava



Polovodiče

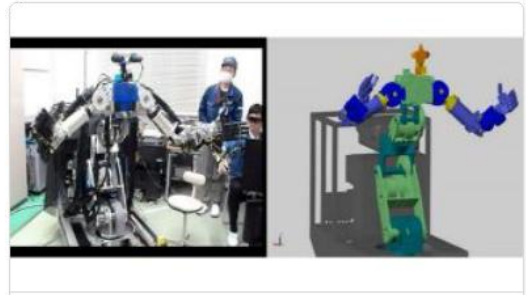
Software a internet

Kde se MATLAB používá ... příklady v oblasti řídicích systémů



The Orion Spacecraft Is Headed to the Moon

Vanderhall Motor Works' Brawley: An All-Electric 303-Horsepower UTV



Advancing Robotics Design with Model-Based Design

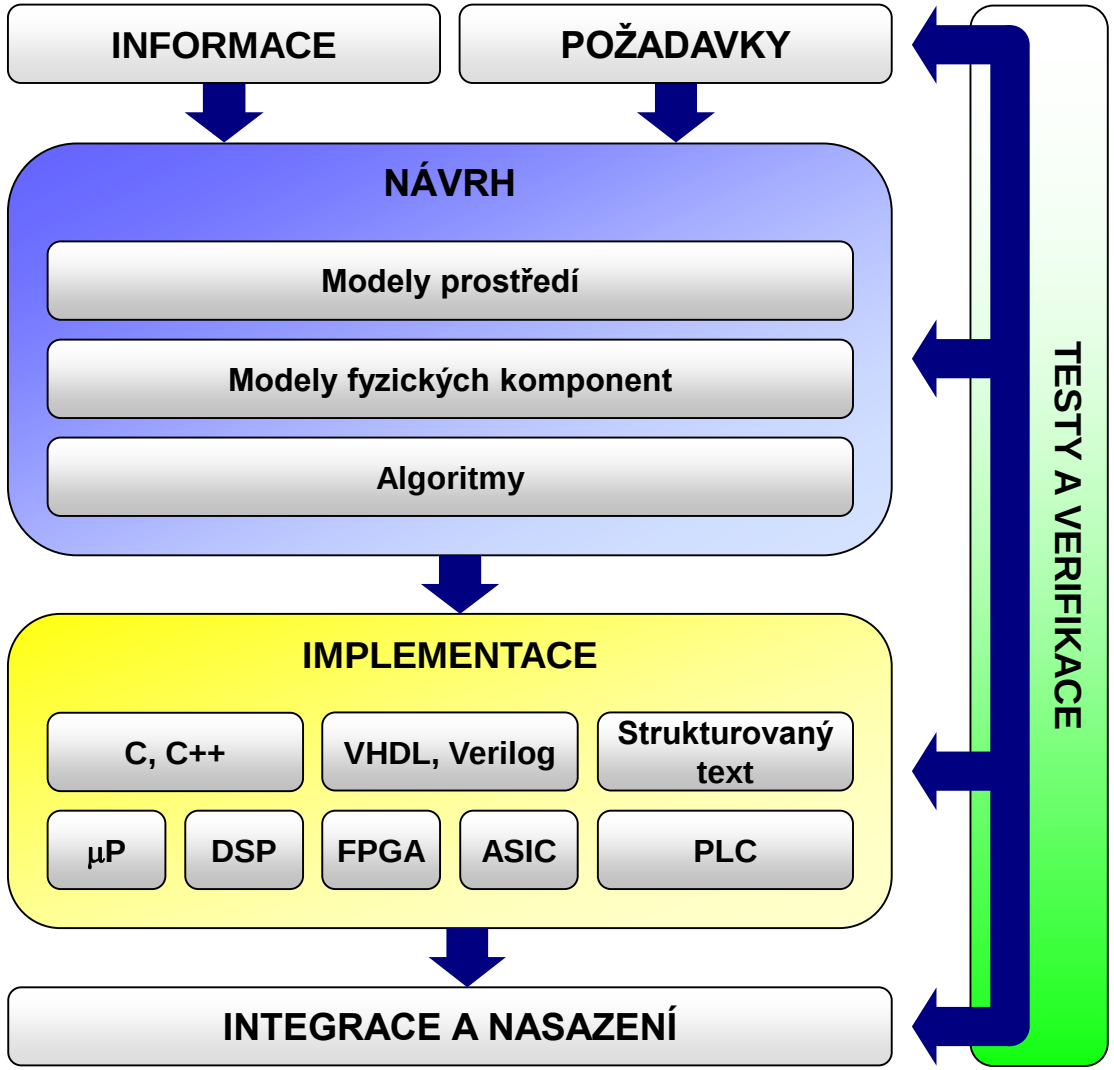
Yanmar Slashes Diesel NOx Emissions with Deep Reinforcement Learning

Dyson Uses Model-Based Design and Automatic Code Generation

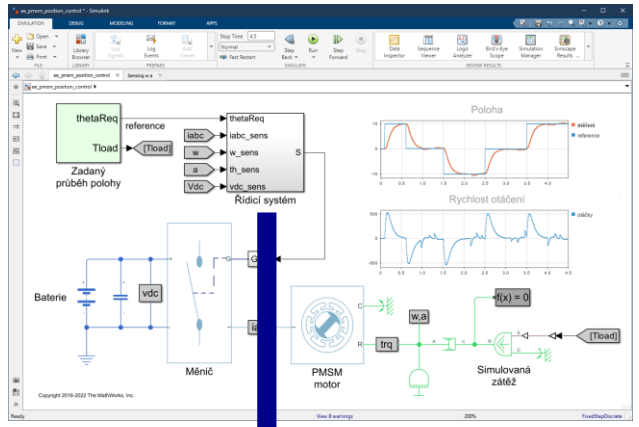
EVLO Energy Storage Accelerates Development of Energy Management...

https://www.mathworks.com/company/customer-stories.html

Vývoj metodou Model-Based Design



Modelování, simulace a testování



Automatické generování kódu

Nasazení

Vývoj metodou Model-Based Design

Systematické využití simulačních modelů napříč vývojovým procesem

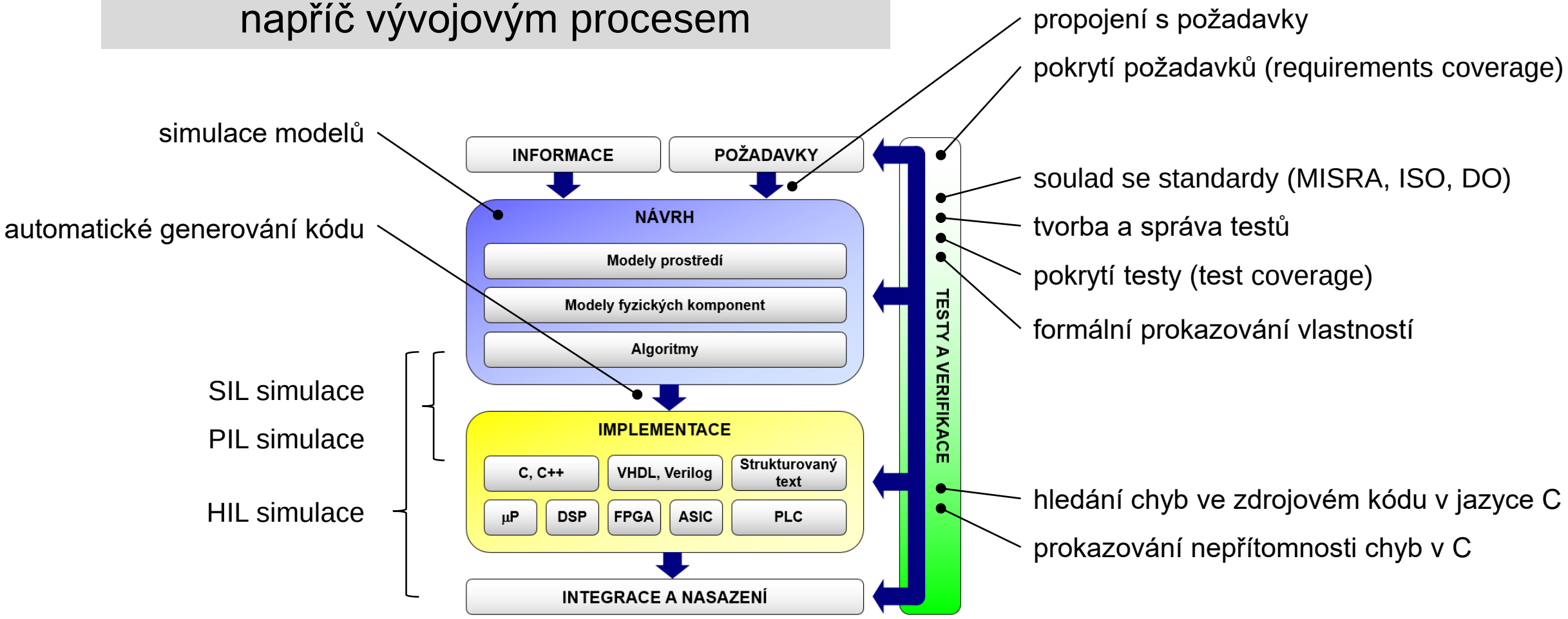


Schéma řídicího systému

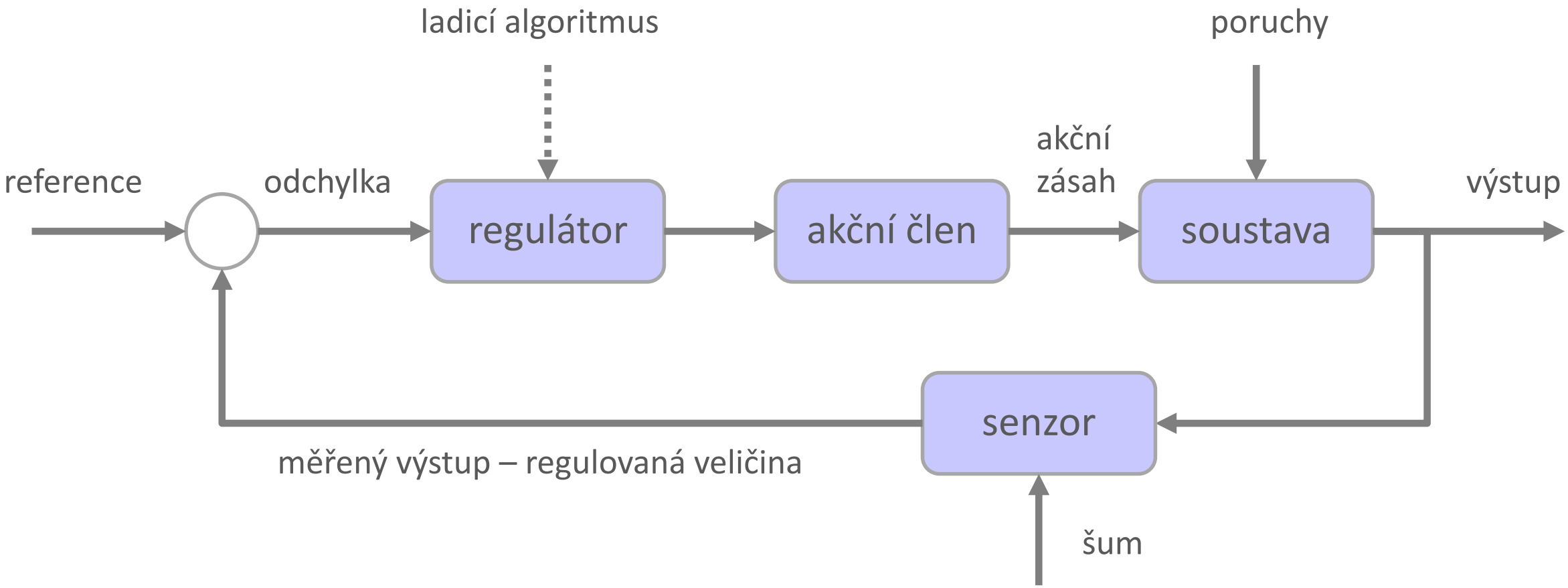
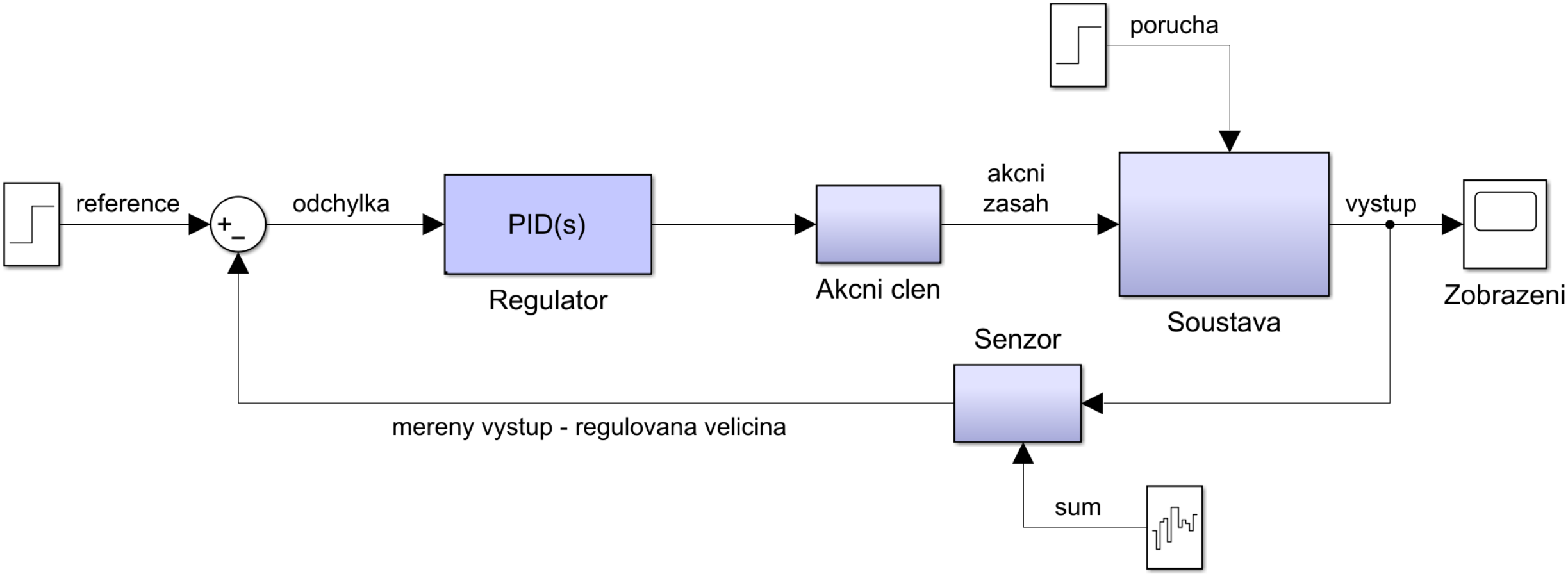
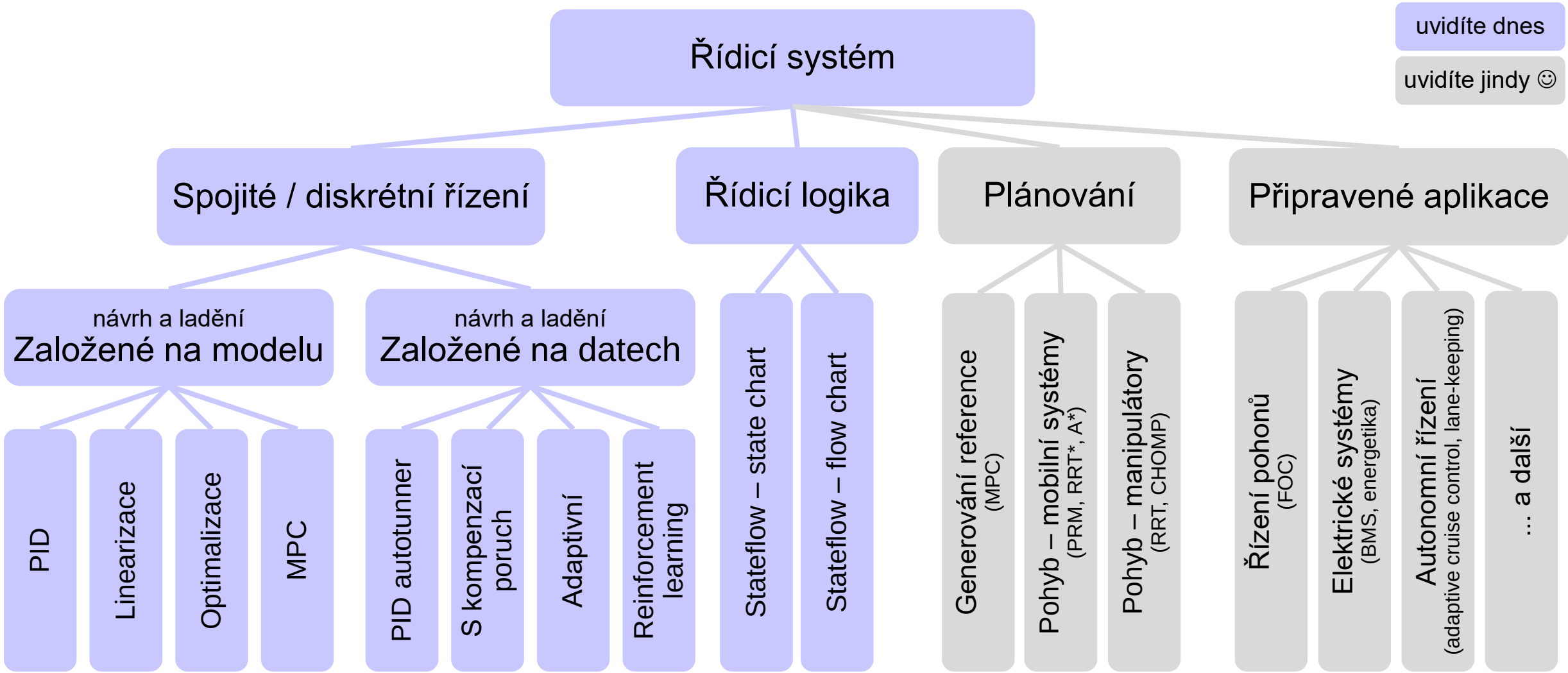


Schéma řídicího systému – Simulink



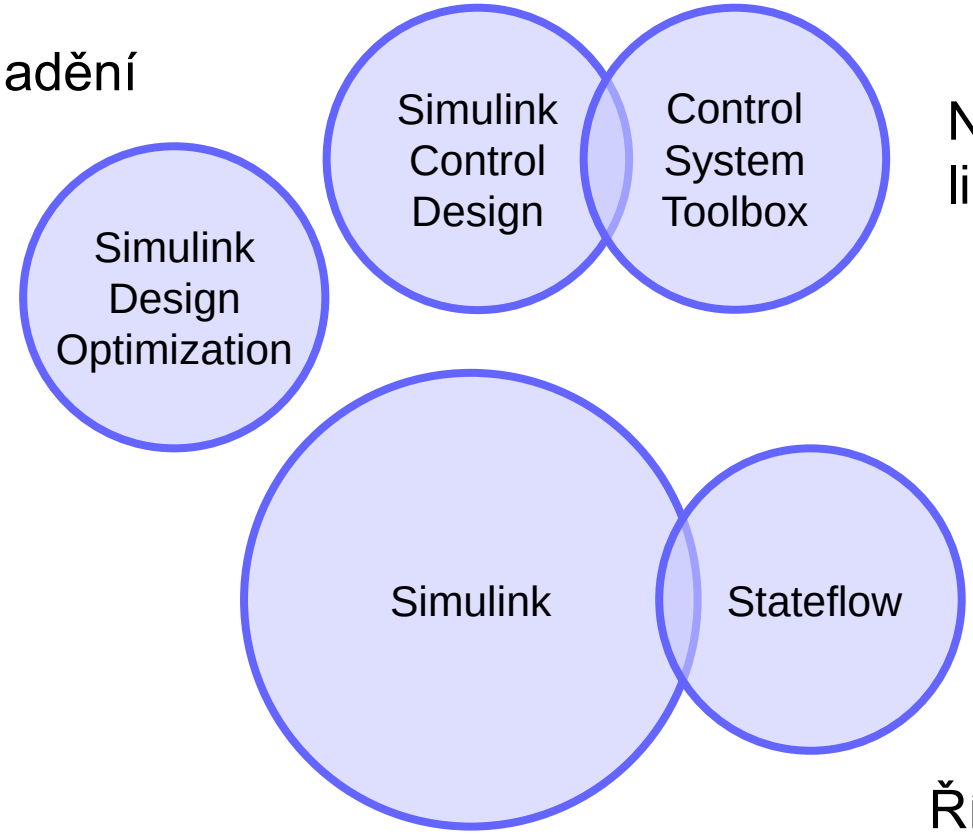
Možnosti návrhu řídicích systémů v prostředí MATLAB

(výběr možností)



MATLAB a Simulink pro návrh řídicích systémů

Optimalizace a ladění parametrů

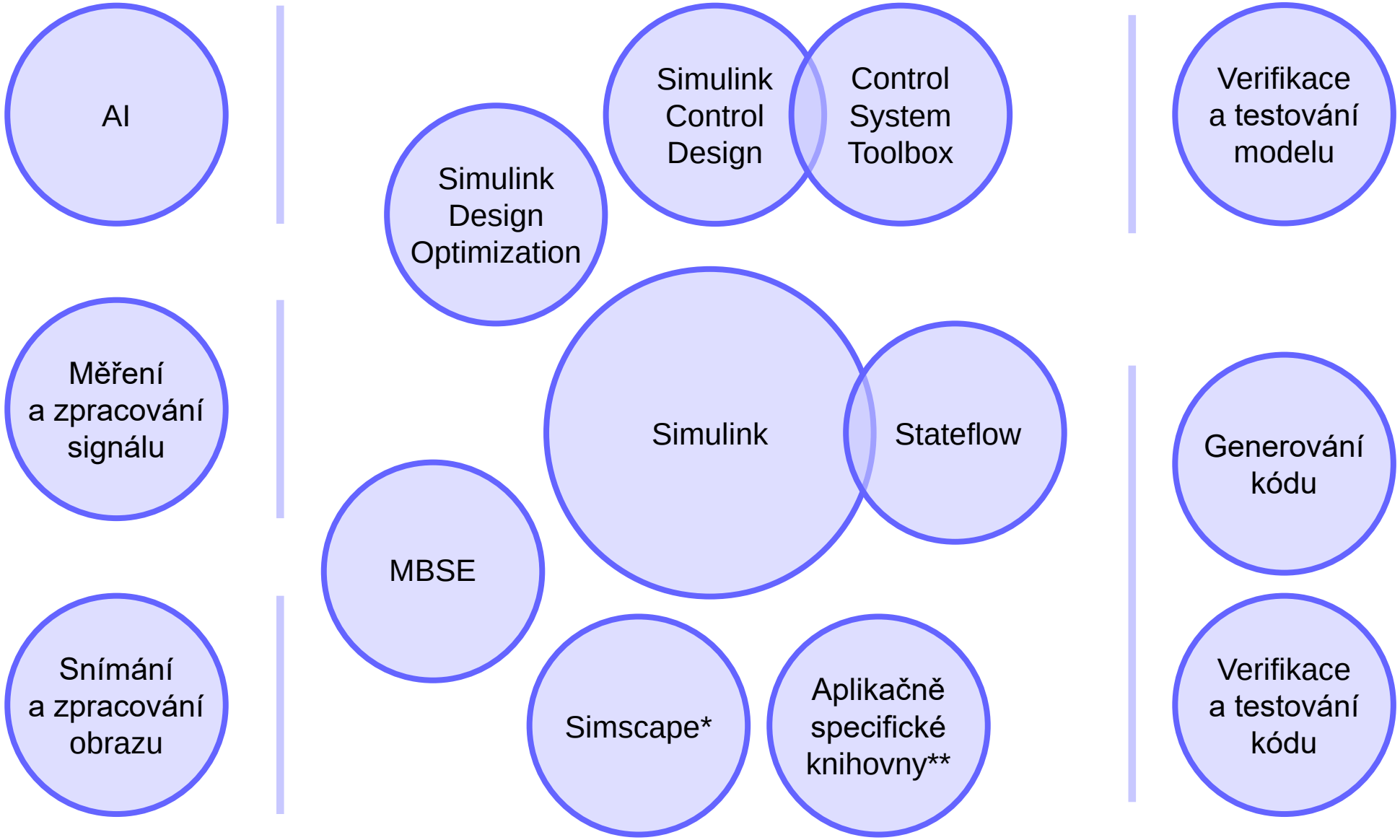


Návrh a ladění řídicích systémů, linearizace modelů

Řídicí a rozhodovací logika

Modelování a simulace řídicích algoritmů a dynamických systémů (soustav)

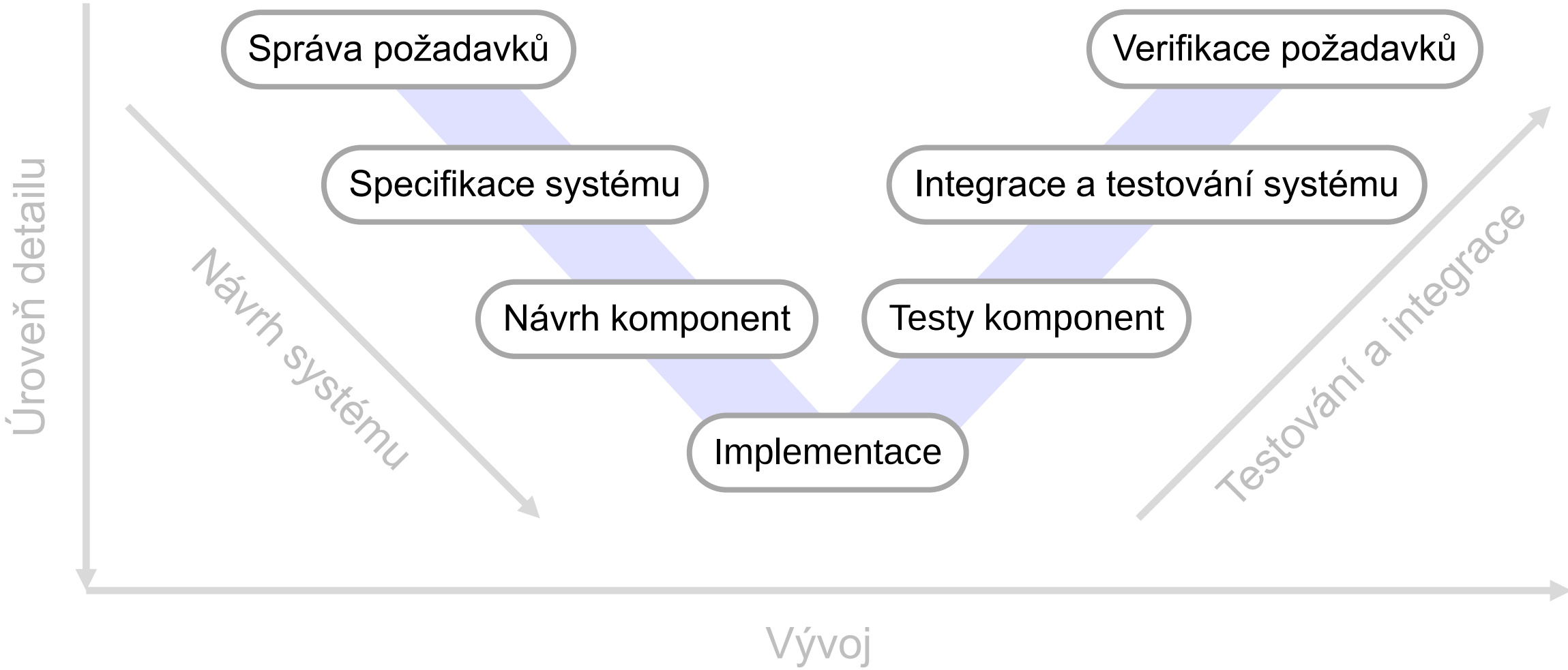
MATLAB a Simulink pro návrh řídicích systémů



*) Electrical
Multibody
Fluids
Battery
Driveline

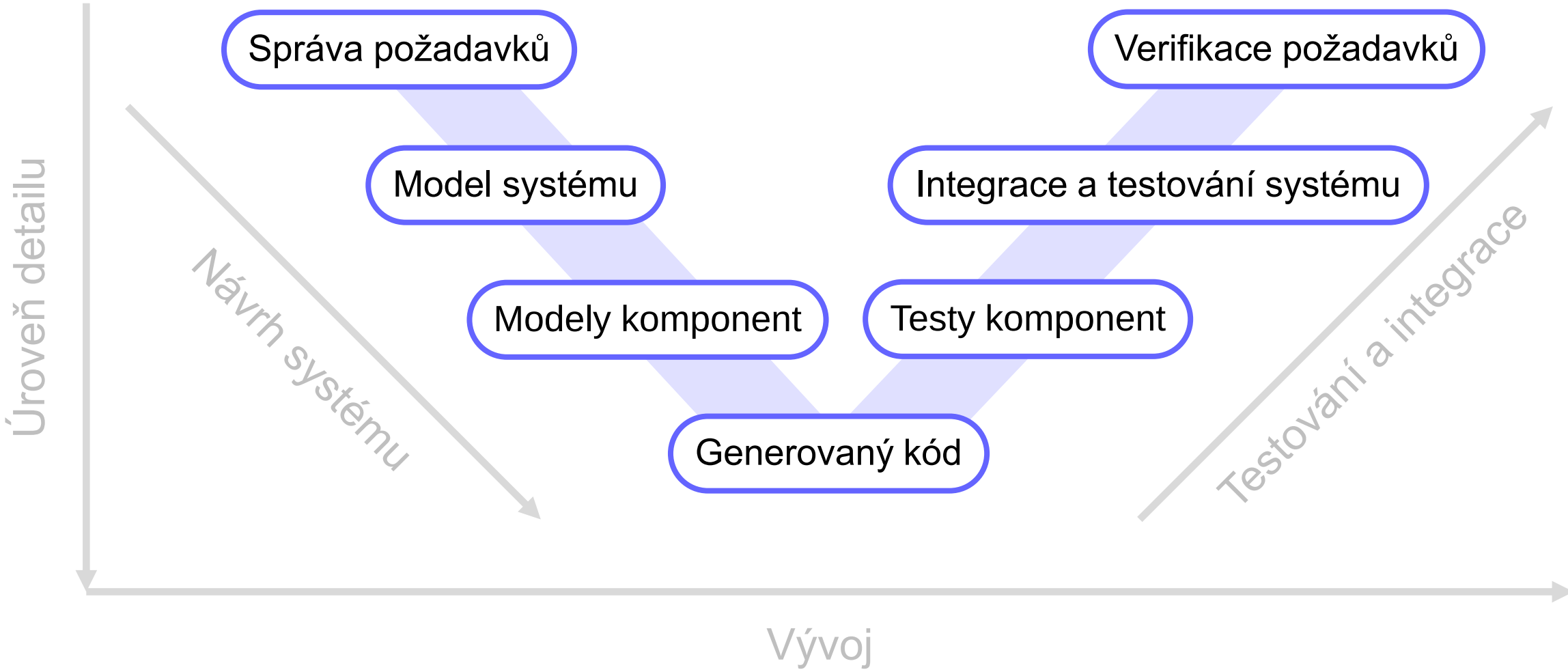
**) Automobily
Letectví
Robotika
Wireless
...

Tradiční vývojový V-cyklus pro (řídicí) systémy



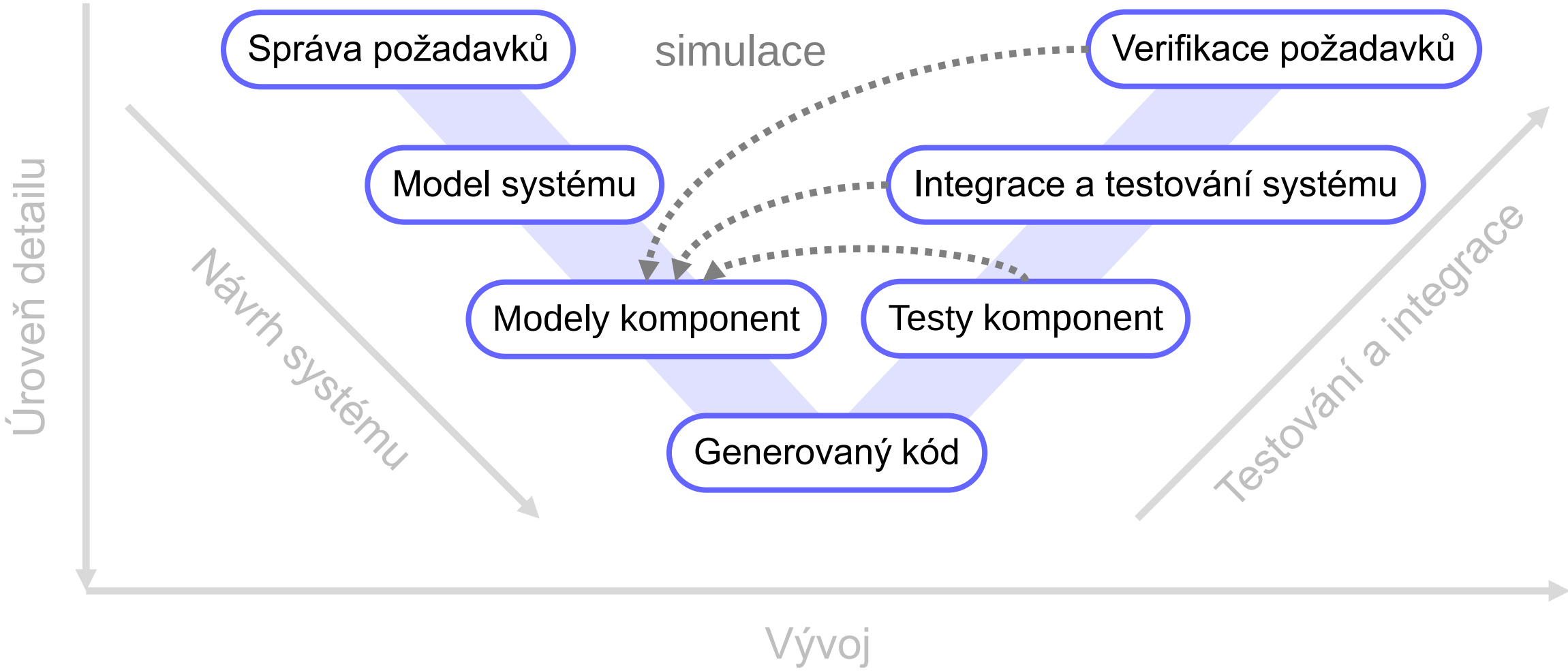
Vývojový V-cyklus v pojetí MBD

Model-Based Design využívá
návrh v podobě modelů a
implementaci pomocí generování kódu



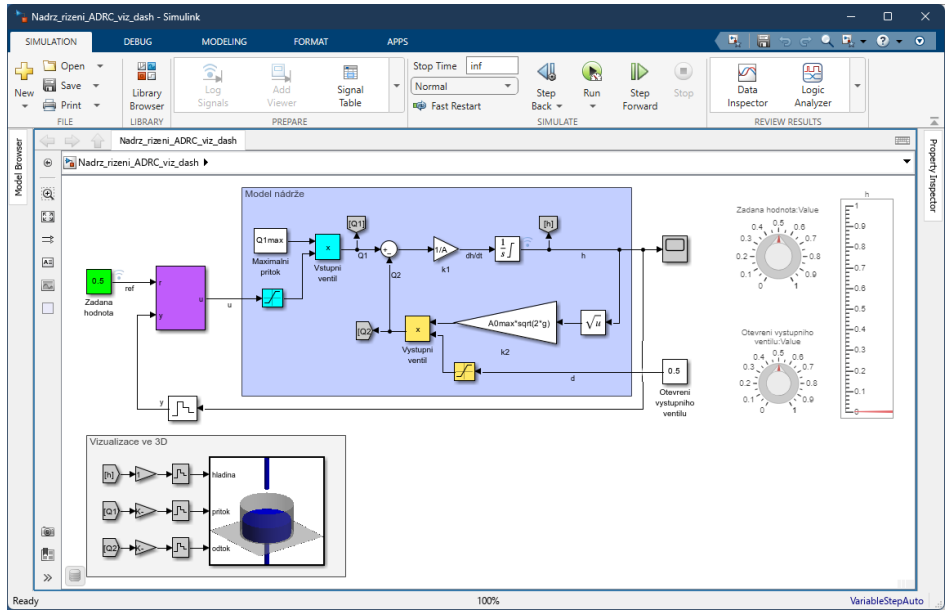
Vývojový V-cyklus v pojetí MBD

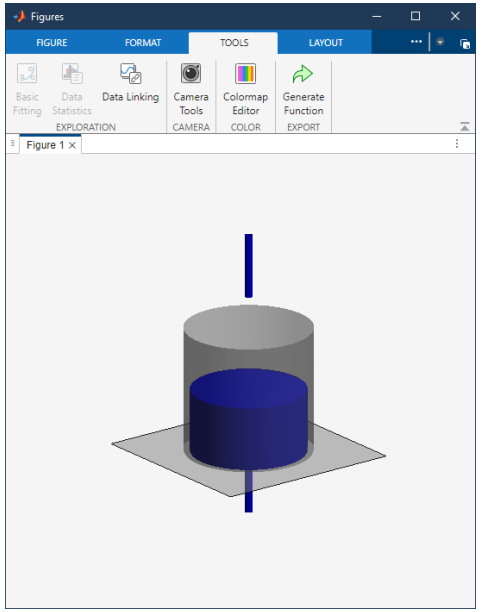
Model-Based Design posouvá část testování (simulace modelů) do levé poloviny diagramu



Ukázka: Regulace hladiny v nádrži

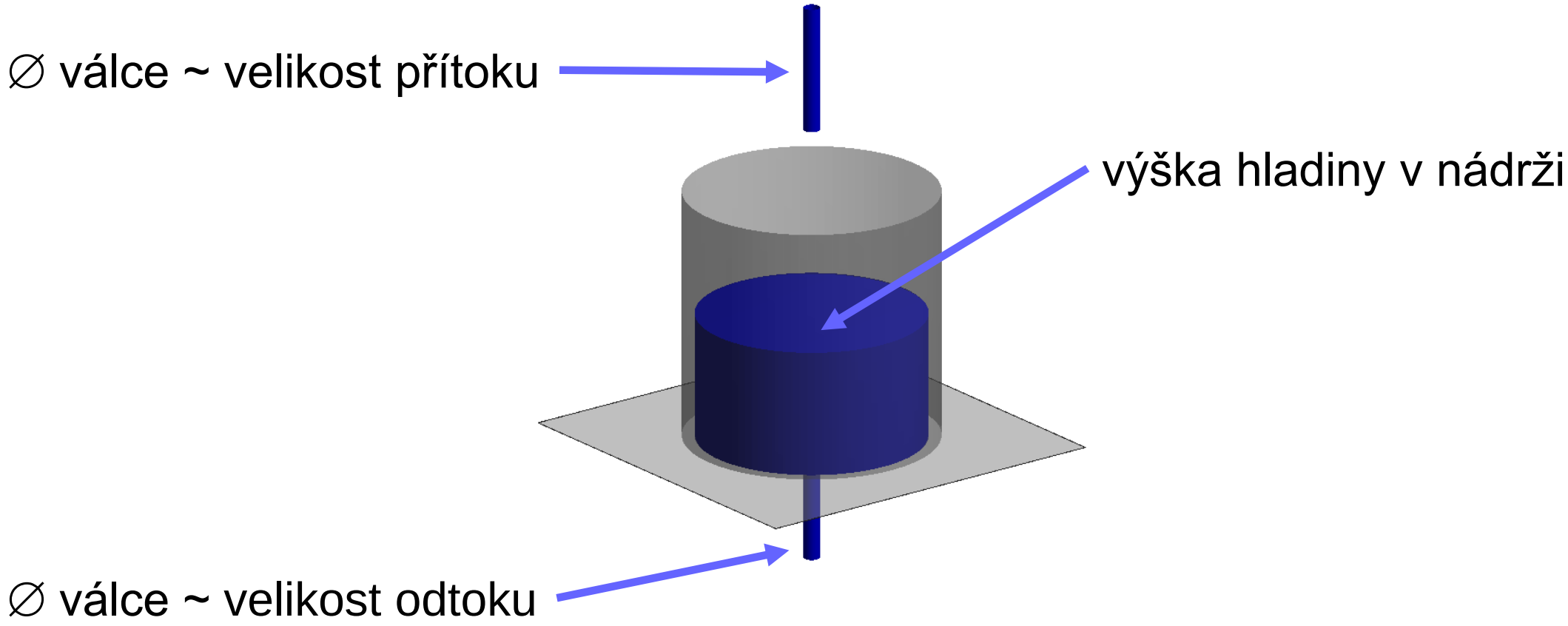
- Nádrž s přítokem a výtokem řízená regulátorem
  - máme nádrž s řízeným přítokem Q1 a variabilním odtokem Q2
  - cíl: zajistit požadovanou výšku hladiny



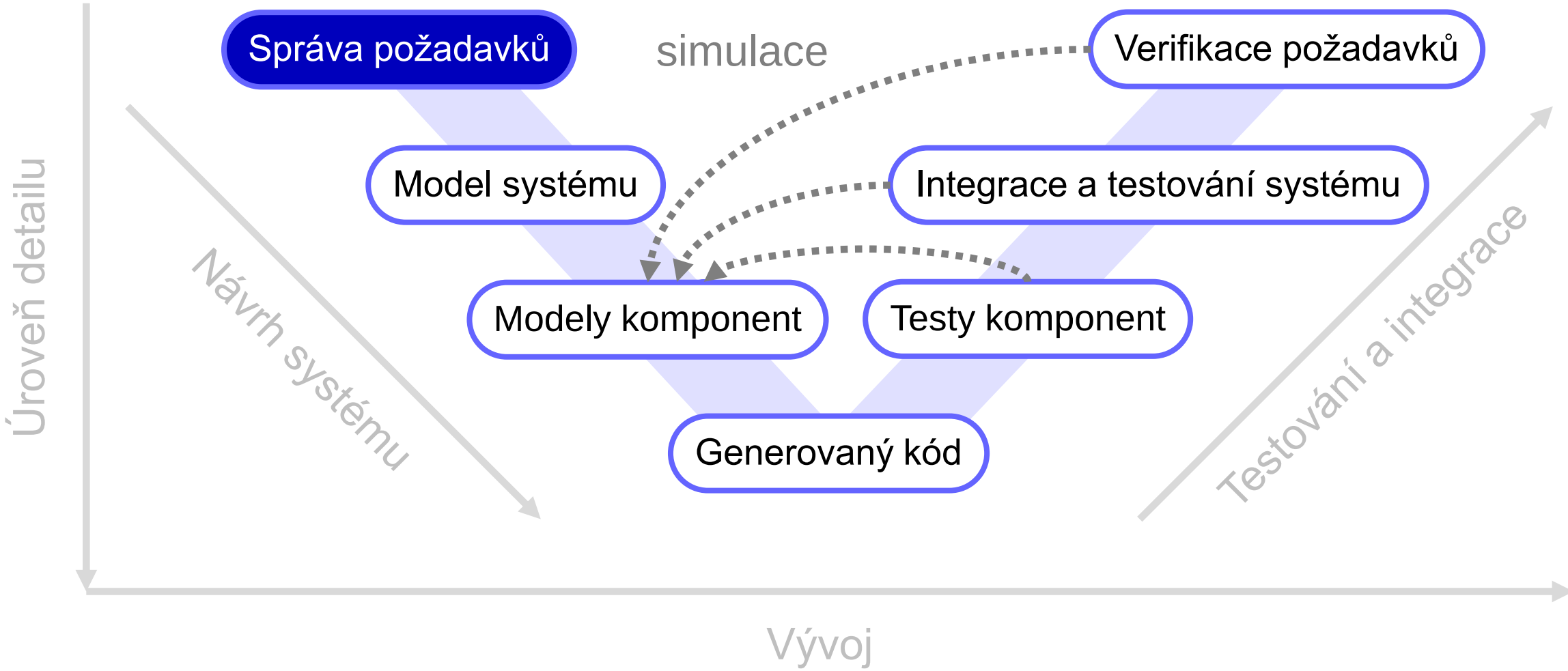


Ukázka: 3D animace

- 3D animace vytvořena pomocí prostředků základního prostředí MATLAB
- Animace volána z modelu pomocí bloku *MATLAB Function*

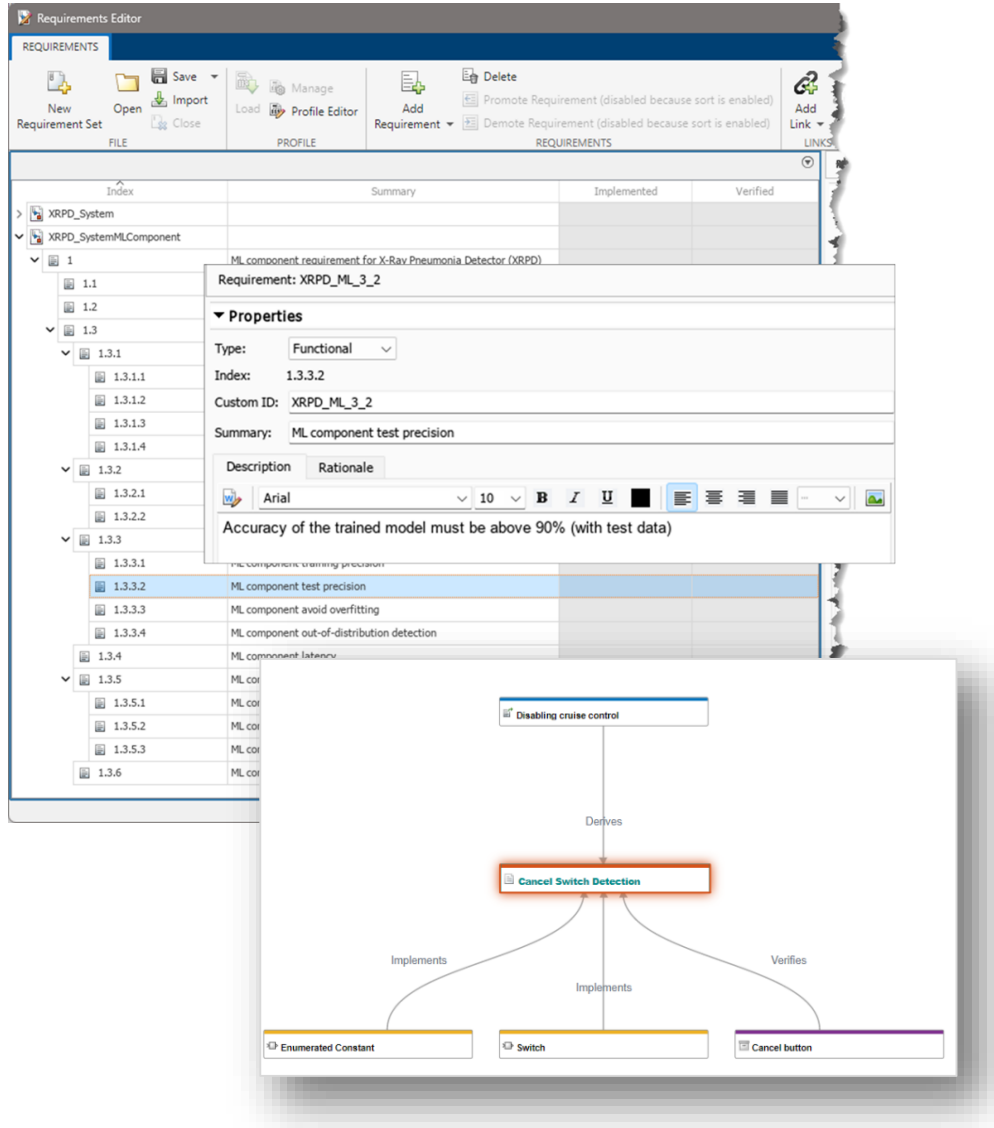


Správa požadavků

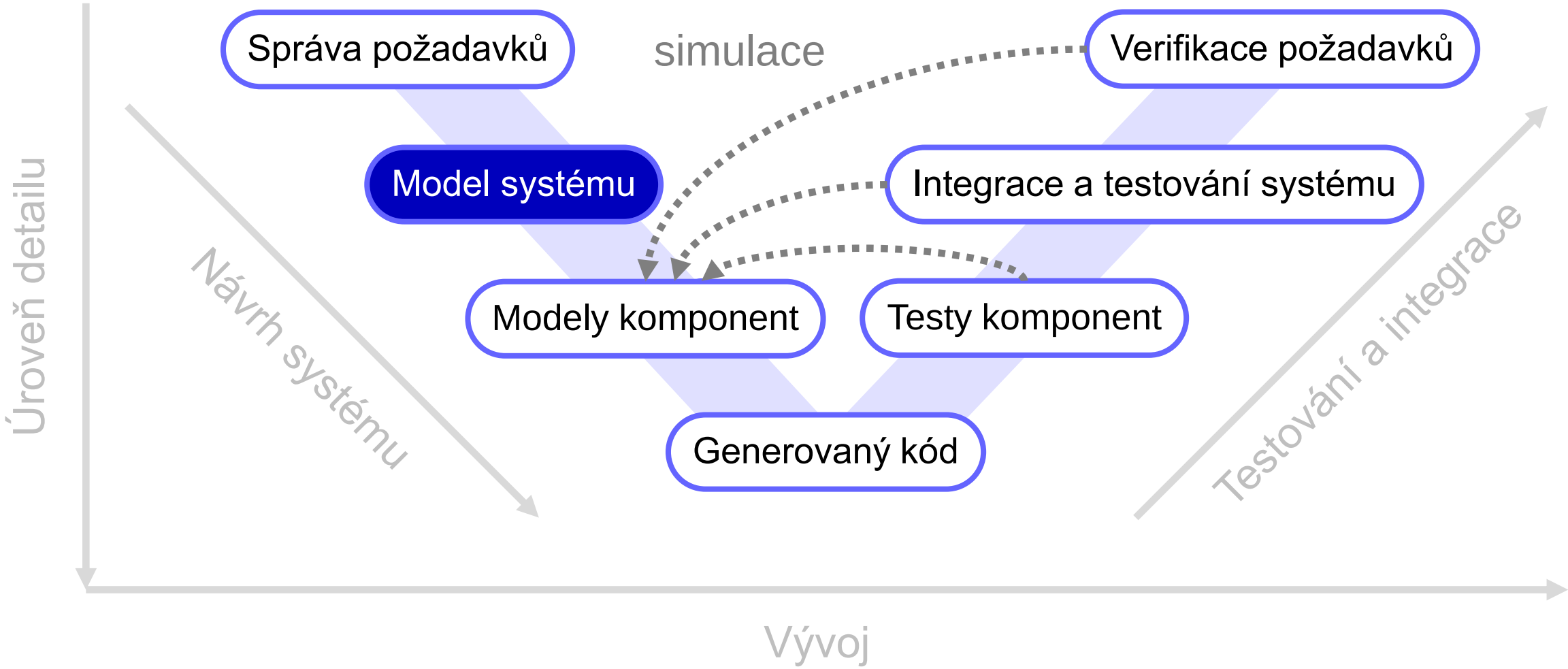


Správa požadavků

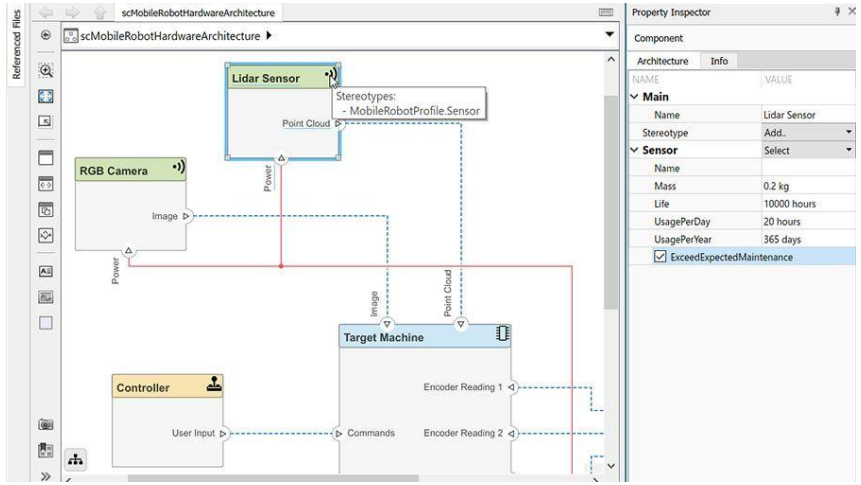
- Požadavky na celkový návrh systému
- Požadavky specifické komponenty
- Requirements Toolbox
  - vytváření, propojení a trasování požadavků
  - Requirements Editor
  - Traceability Matrix, Traceability Diagrams
  - propojení/import: IBM Doors, MS Word, ReqIF
- Klíčové otázky
  - jsou všechny požadavky implementovány?
  - jak budou požadavky testovány?
  - lze vysvětlit chování modelu?



Modelování systému ~ návrh architektury

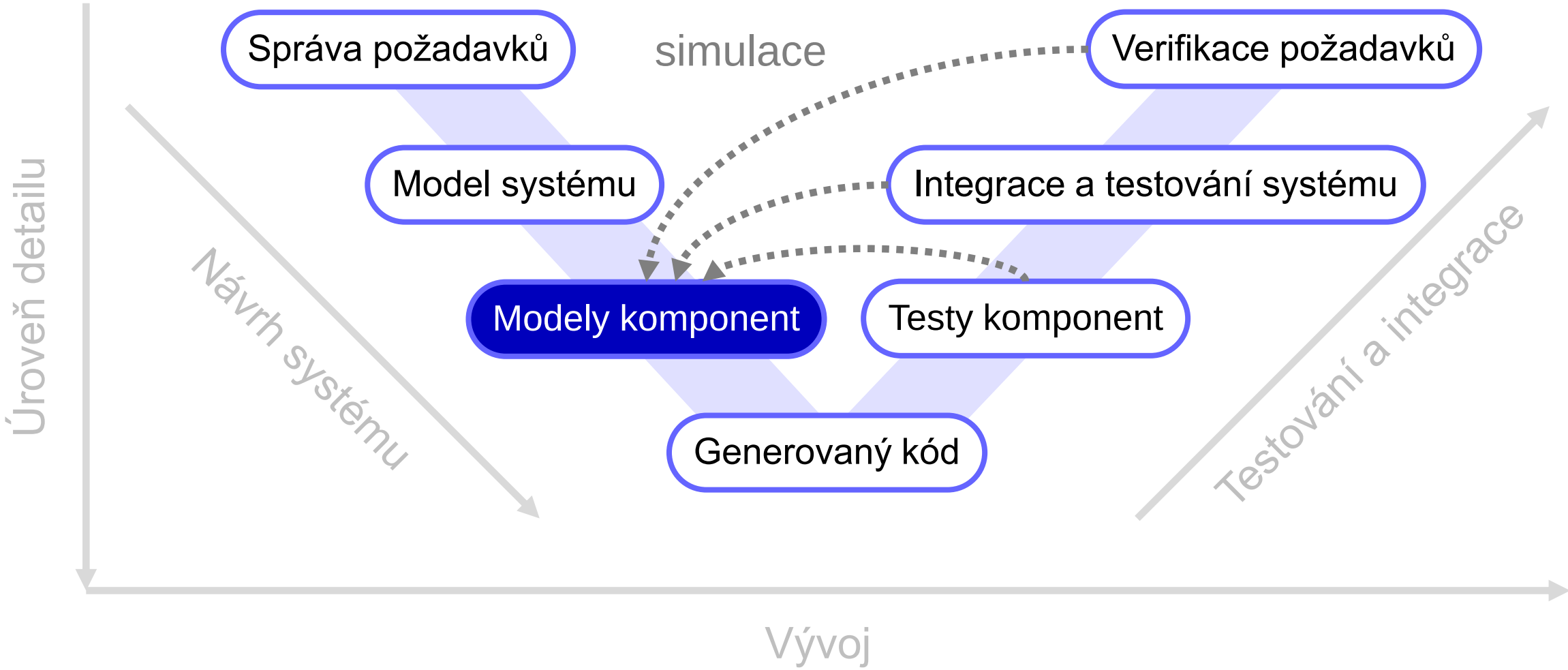


Návrh architektury systému



- System Composer
  - nástroj pro návrh a dekompozici architektury systému
  - komponenty, rozhraní, stereotypy, pohledy, ...
  - propojení s požadavky
  - dále v návrhu lze provázat s modely komponent
- Simulink (bez nástroje System Composer)
  - architekturu představuje kořenová úroveň modelu systému
  - počáteční rozvržení architektury pomocí prázdných subsystémů

Modelování komponent ~ návrh funkčnosti



Modelování soustav: Přístupy k modelování

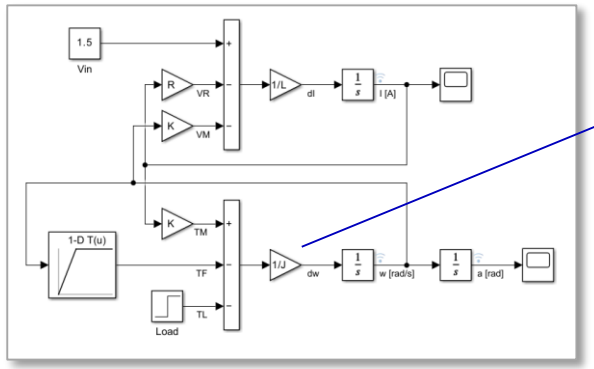
- Pro různé situace jsou vhodné různé přístupy

Fyzikální vztahy **Naměřená data**

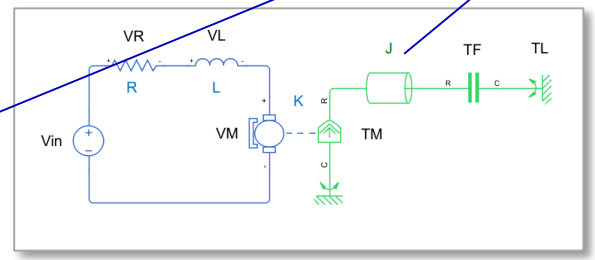
„white-box“ model „grey-box“ model „black-box“ model

Modelování rovnic

$$L\frac{dI}{dt} = V_{in} - RI - K\omega \qquad J\frac{d\omega}{dt} = KI - f(\omega) - T_L$$

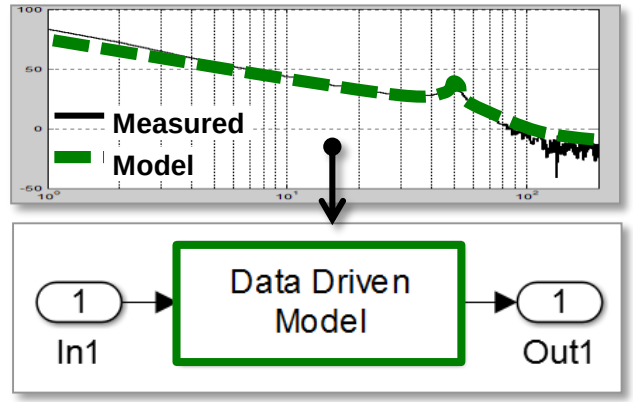


Fyzikální sítě



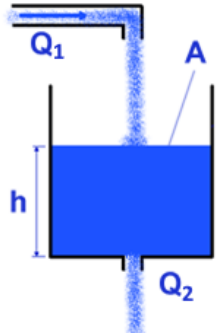
Ladění neznámých parametrů

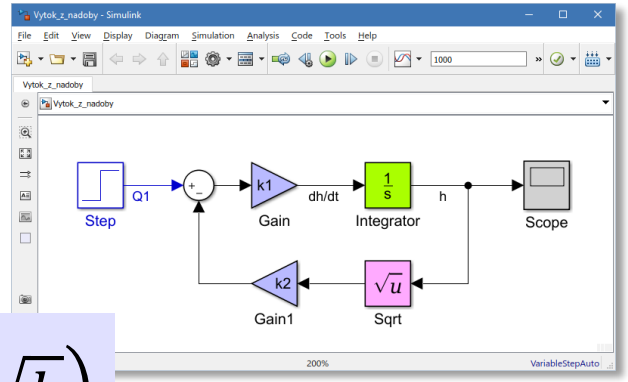
Identifikace soustav



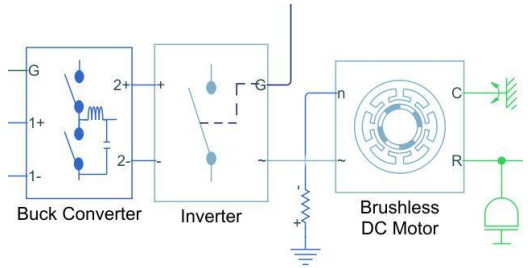
Modelování soustav: Prostředky

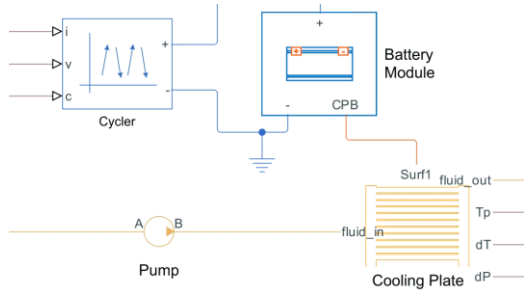
- Popis matematickými rovnicemi (Simulink)
  - matematicko-fyzikální analýza
  - identifikace soustavy z naměřených dat
- Fyzikální modelování (Simscape)
  - elektrické a elektromechanické systémy
  - baterie
  - tekutinové systémy
  - 3-D mechanika
  - převodové systémy
- Aplikačně zaměřené
  - automobily, letadla, drony, roboty

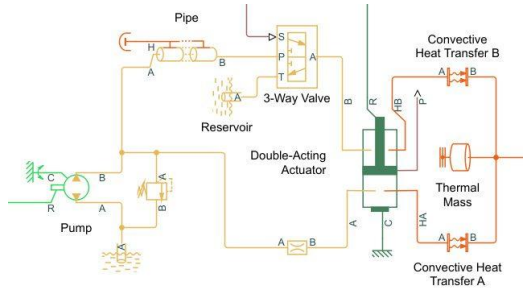


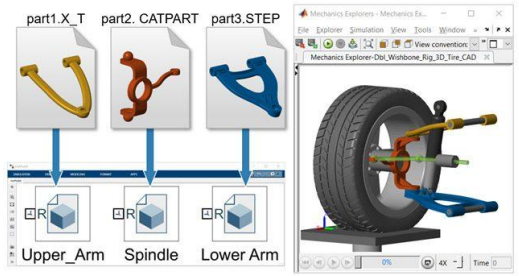


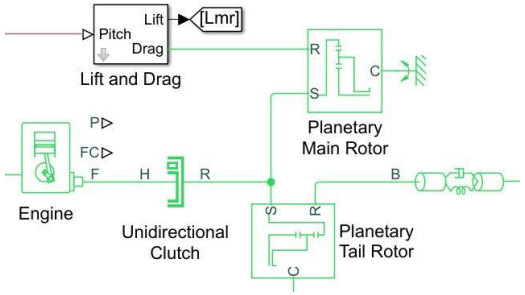
$$\frac{dh}{dt} = k_1\left(Q_1 - k_2\sqrt{h}\right)$$











Ukázka: Jak získat model nádrže?

- Odvození rovnice

$$\frac{dV}{dt} = Q_1 - Q_2 \qquad V = A\,h \qquad Q_2 = A_0\sqrt{2gh}$$

$$A\frac{dh}{dt} = Q_1 - A_0\sqrt{2gh}$$

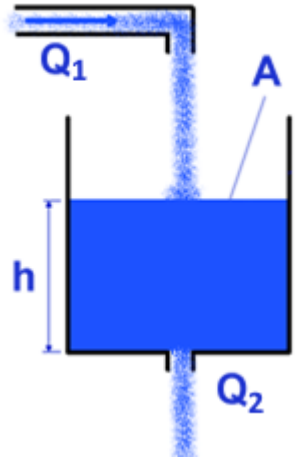
$$\frac{dh}{dt} = \frac{1}{A}\left(Q_1 - A_0\sqrt{2g}\sqrt{h}\right) \qquad k_1 = \frac{1}{A} \qquad k_2 = A_0\sqrt{2g}$$

- Tvar vhodný pro modelování v Simulinku

$$\frac{dh}{dt} = k_1\left(Q_1 - k_2\sqrt{h}\right)$$

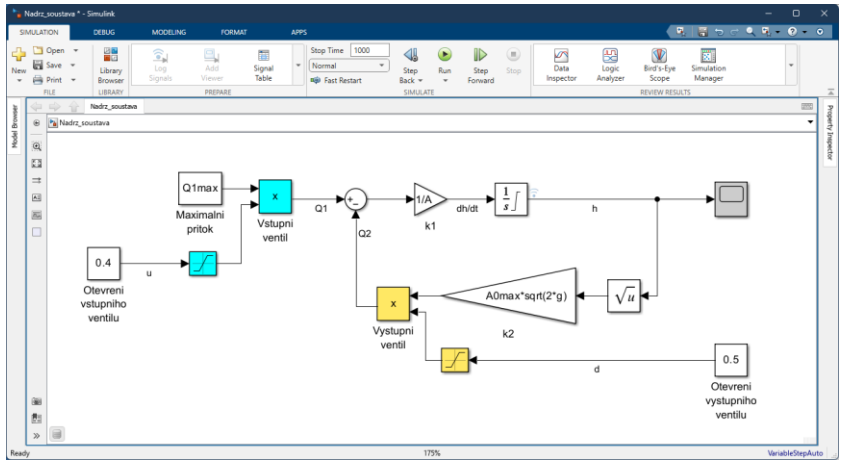
$$Q_1 = Q_{1\max}\,k_{vstup}$$

$$A_0 = A_{0\max}\,k_{výstup}$$



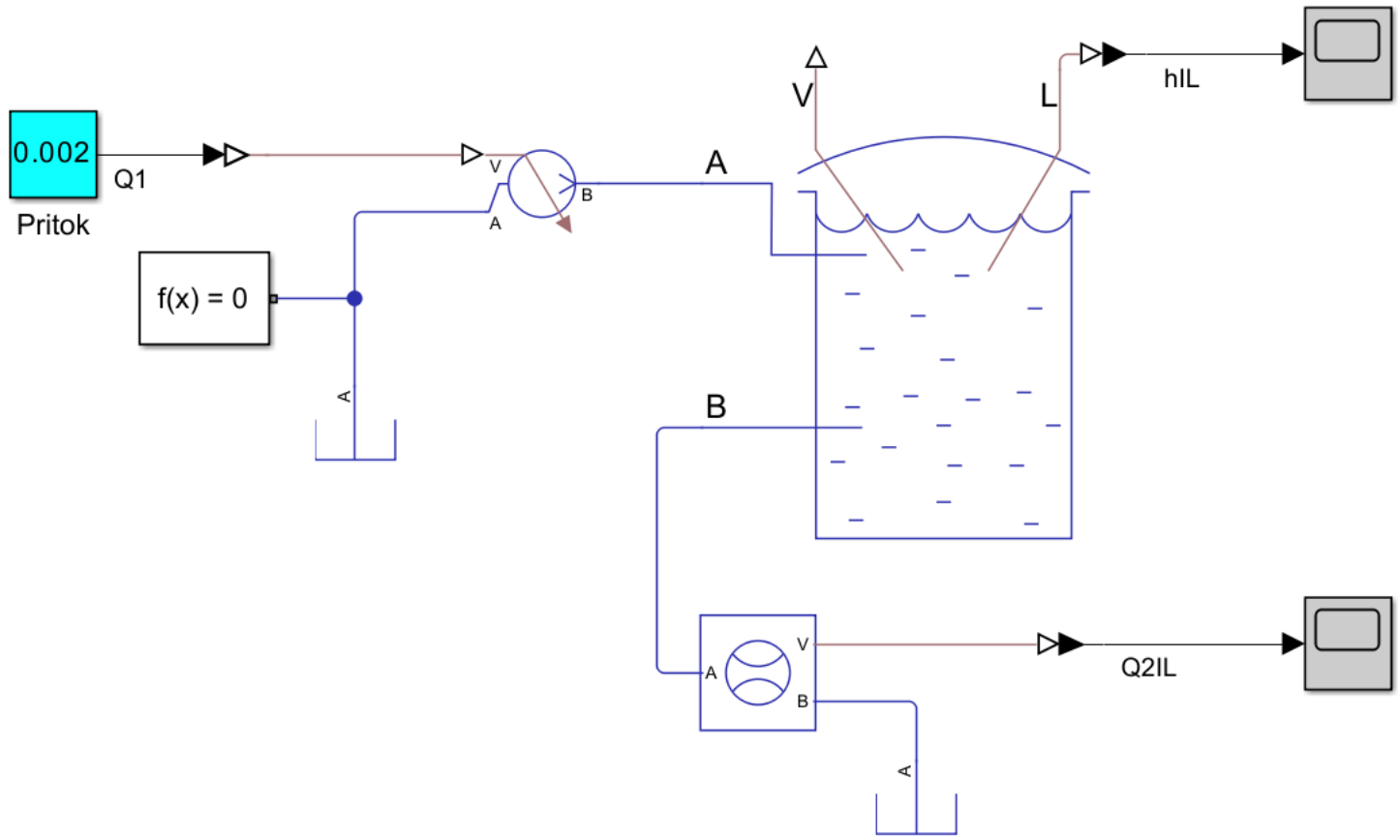
Hodnoty parametrů:

- A = 0.2 m^2
- A_{0max} = 0.001 m^2
- Q_{1max} = 0.005 m^3/s



Ukázka: Jak získat model nádrže?

- Nebo použít knihovny Simscape



Hodnoty parametrů:

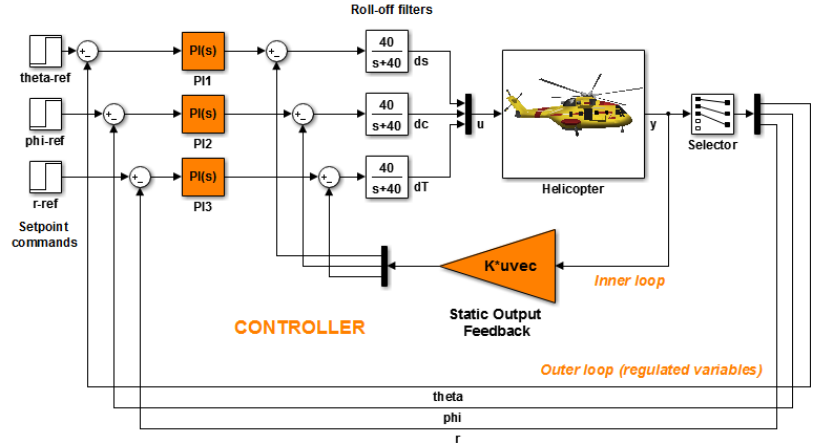
- $A = 0.2\ m^2$
- $A_{0max} = 0.001\ m^2$
- $Q_{1max} = 0.005\ m^3/s$

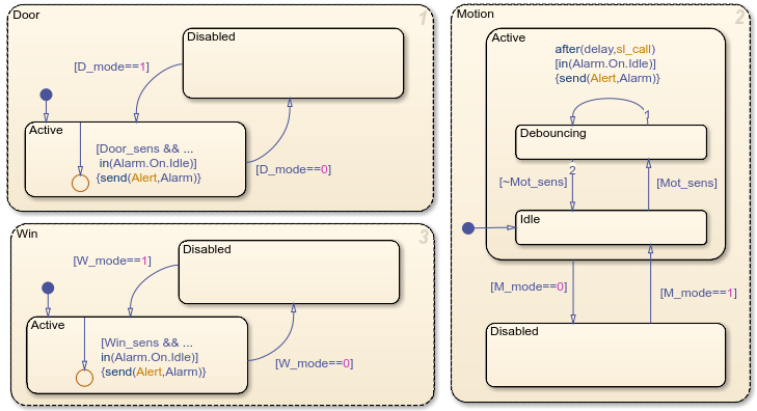
Modelování algoritmů

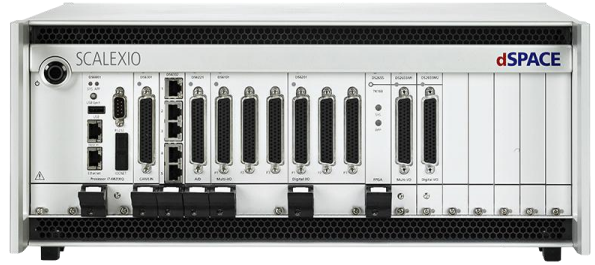
- Řídicí systémy
- Logické a rozhodovací systémy
- Zpracování signálu a komunikace
- Zpracování obrazu a počítačové vidění
- a další ...

- Společná simulace soustav a algoritmů

- Generování kódu pro cílové platformy
  - C/C++, HDL, PLC (strukturovaný text), CUDA
  - simulace a testování v reálném čase, produkce



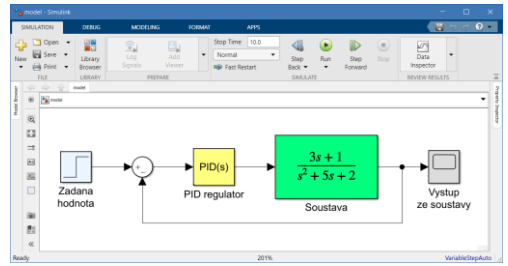


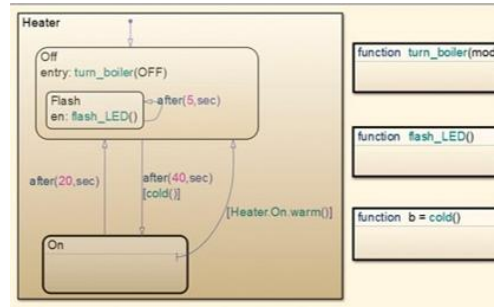


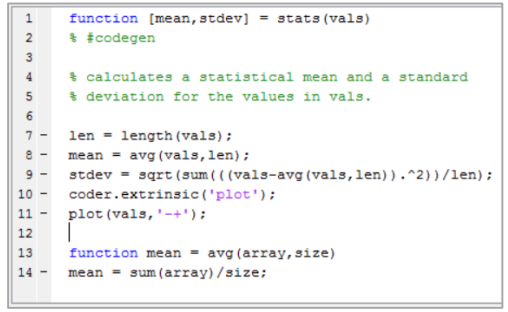
Modelování algoritmů: Prostředky

Model v prostředí Simulink

- Výpočetní a dynamické algoritmy **Simulink**
- Stavové a logické algoritmy **Stateflow**
- Kód **MATLAB / C**



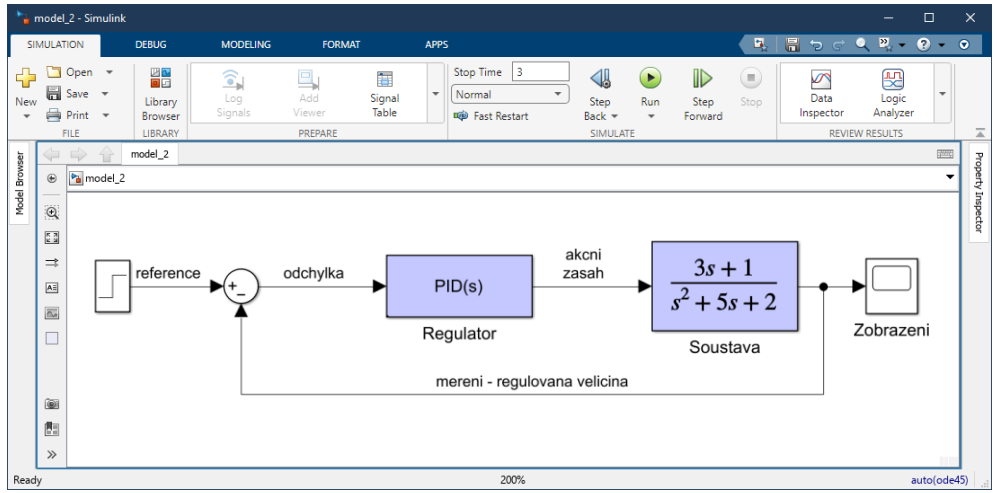


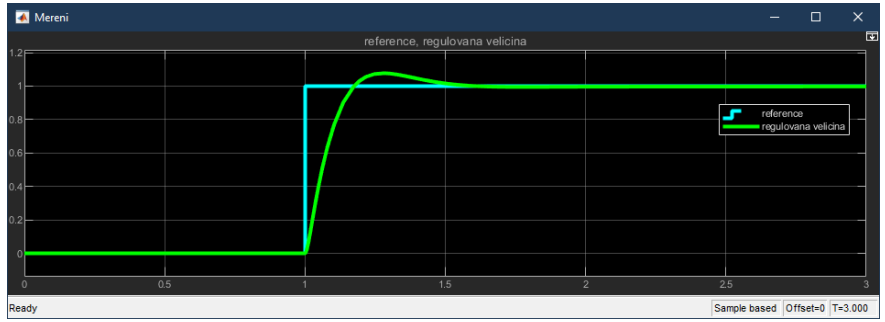


```
function [mean,stdev] = stats(vals)
% #codegen

% calculates a statistical mean and a standard
% deviation for the values in vals.

len = length(vals);
mean = avg(vals,len);
stdev = sqrt(sum(((vals-avg(vals,len)).^2))/len);
coder.extrinsic('plot');
plot(vals,'--+');

function mean = avg(array,size)
mean = sum(array)/size;
```

Návrh řídicích systémů v prostředí MATLAB a Simulink

- Propojení modelů soustav s regulátory
  - spojité a diskrétní prvky v jednom modelu
  - bohaté knihovny vstupních signálů
- Libovolná architektura řídicích systému
  - blok PID regulátoru v mnoha variantách
  - spojité, diskrétní, stavové regulátory
  - adaptivní a prediktivní řízení
- Nástroje pro ladění řídicích systémů

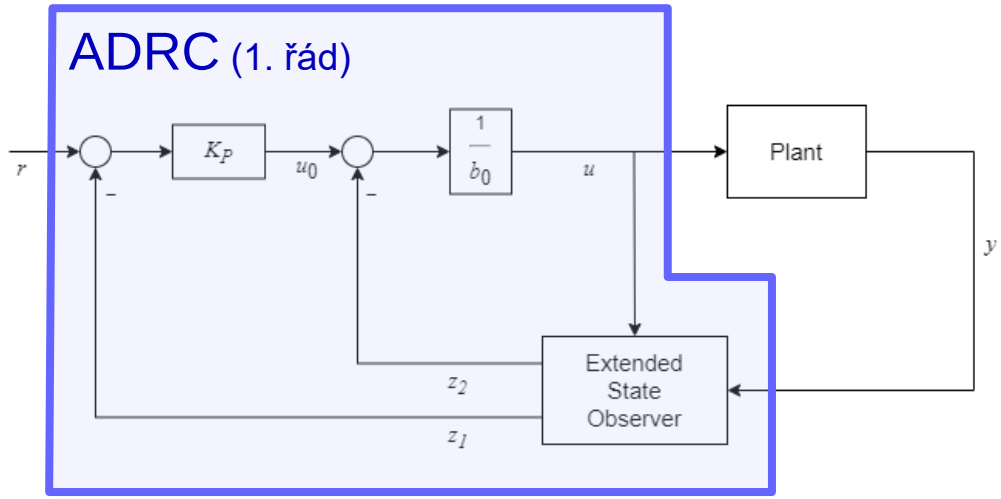




Active Disturbance Rejection Control (ADRC)

- Řízení soustav s neznámou dynamikou a vnitřními a vnějšími poruchami
  - regulace s robustním potlačení poruch a minimálním překmitem
- Blok *Active Disturbance Rejection Control*
- Princip ADRC
  - nevyžaduje model soustavy
  - využívá jednoduchou aproximaci známé dynamiky

$$\dot{y}(t) = b_0 u(t) + f(t) \quad \text{nebo} \quad \ddot{y}(t) = b_0 u(t) + f(t)$$

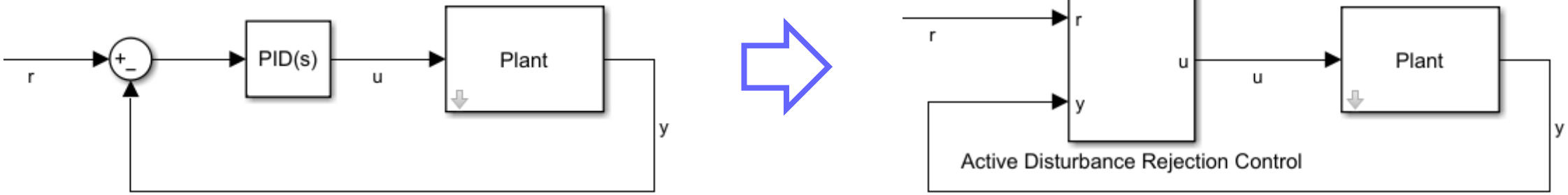


  - neznámá dynamika a poruchy $f(t)$ modelovány jako rozšířený stav soustavy

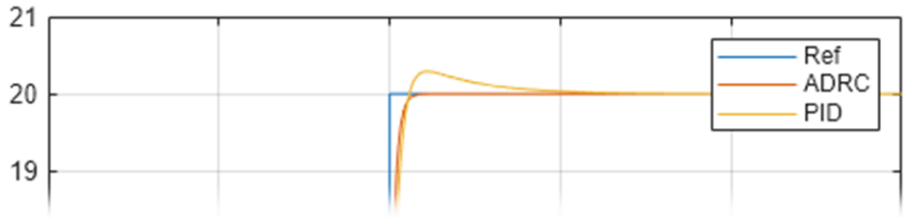
https://www.mathworks.com/help/slcontrol/ug/active-disturbance-rejection-control.html

Active Disturbance Rejection Control (ADRC)

- Může sloužit jako alternativa PID regulace
  - zejména pro výkonovou elektroniku a řízení motorů
  - obdobné výpočetní nároky jako PID

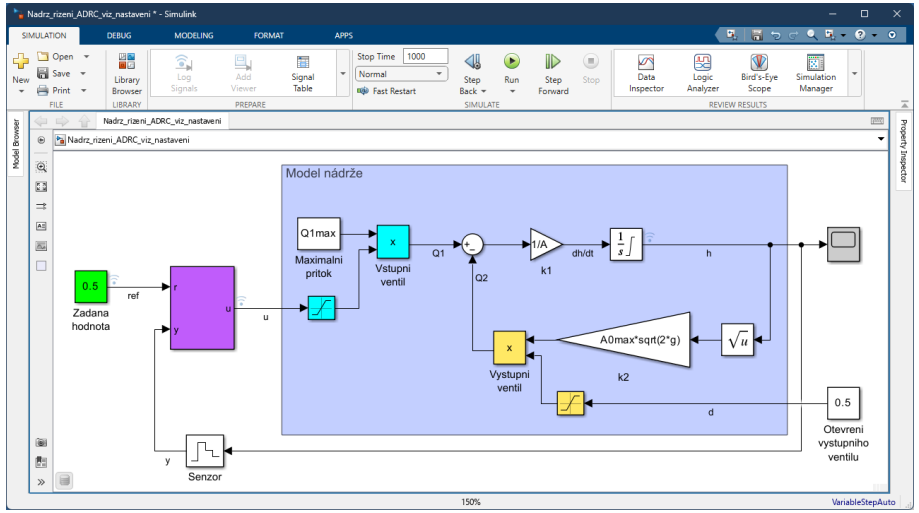


- Jednoduchý postup ladění
  - pro širokou škálu provozních podmínek



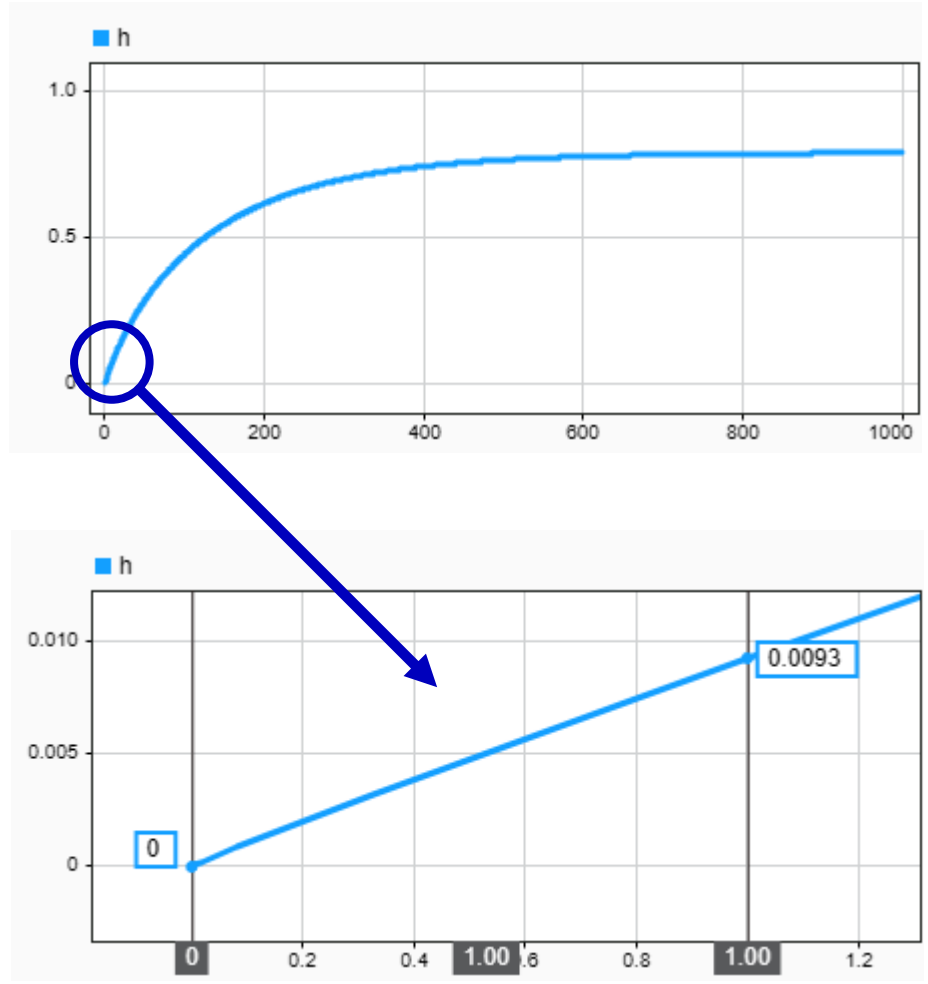
Ukázka: Zapojení a nastavení ADRC regulace

- Zadána požadovaná hodnota výšky hladiny
  - konstantní hodnota $ref = 0.5$
- Zapojena zpětnovazební regulace
  - blok *Active Disturbance Rejection Control*
  - není explicitní výpočet regulační odchylky
- Simulace chování senzoru výšky hladiny
- Nastavení parametrů ADRC regulátoru
  - dle postupu uvedeného v dokumentaci



Ladění ADRC regulátoru (1.řád)

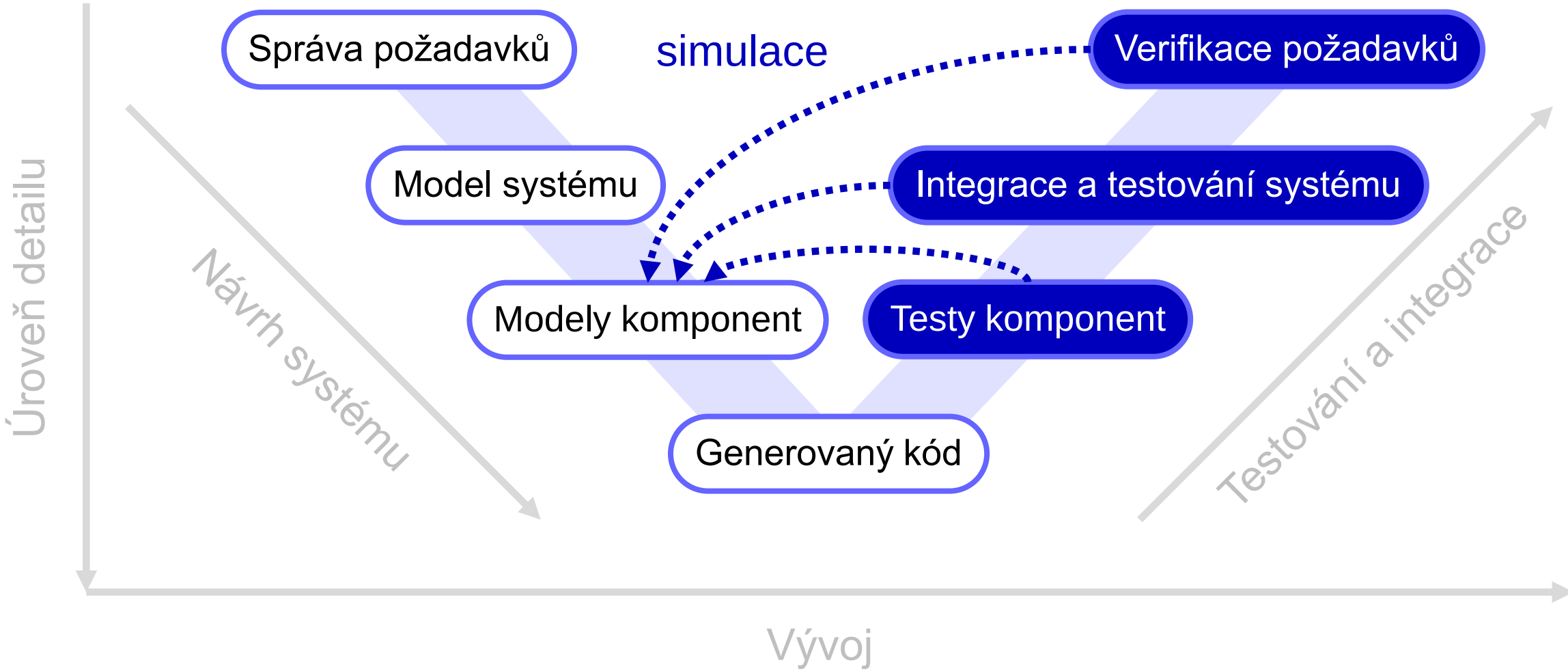
- Určení kritického zesílení b_0
  - experiment ~ přechodová odezva soustavy
  - proměření počátku odezvy
  - $a = \frac{\Delta y}{\Delta t}$, $b_0 = \frac{a}{\Delta u}$
- Šířka frekvenčního pásma regulátoru
  - dle zadaných požadavků na chování systému
- Šířka frekvenčního pásma pozorovatele
  - 5 až 10 krát vyšší
- Hotovo ...



(pro model 2. řádu obdobně)

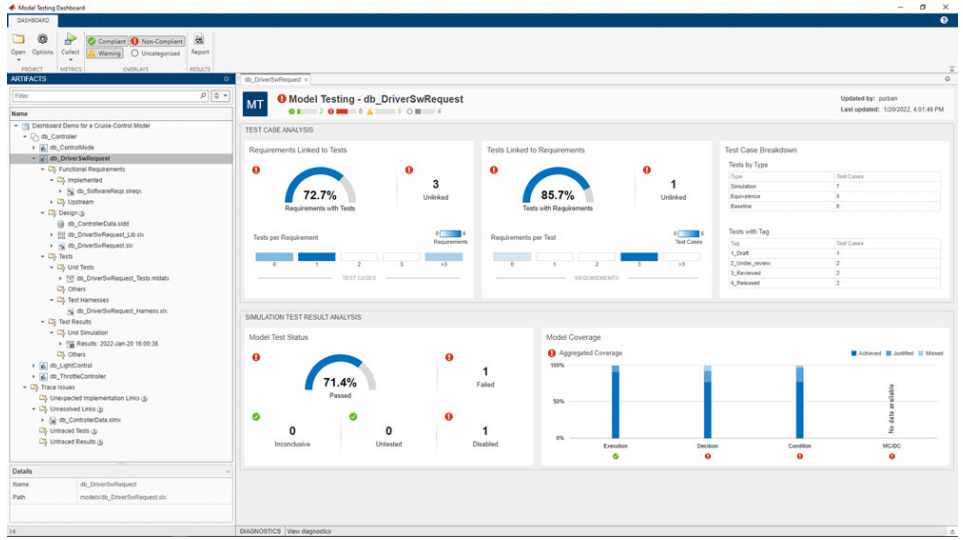
https://www.mathworks.com/help/slcontrol/ug/design-adrc-for-watertank.html

Integrace a testování modelu systému a jeho komponent



Testování a verifikace modelů

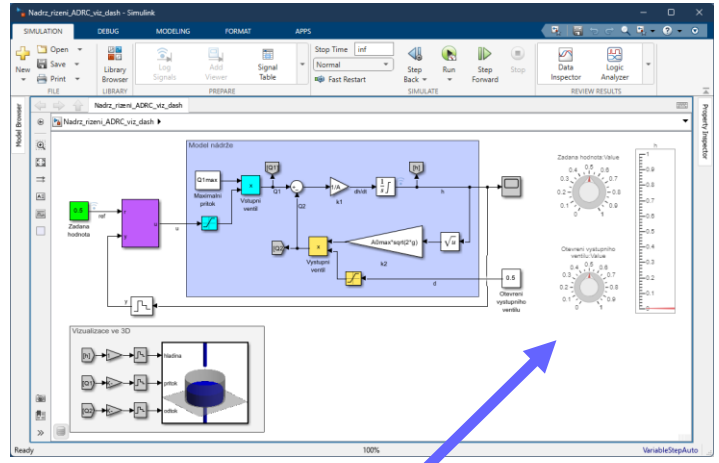
- Verifikace navrženého algoritmu v podobě simulačního modelu
  - ověření návrhu a chování algoritmu pomocí simulace (model-in-the-loop)
- Systematické testování prostřednictvím simulací
  - nástroje pro tvorbu a správu testů a verifikace požadavků (Test Harness, Test Manager)
  - analýza pokrytí testy (test coverage)
- Kontrola souladu s průmyslovými standardy
  - MISRA, ISO 26262, DO-178C, DO-331, ...
- Formální prokazování vlastností
- Verifikace požadavků na úrovni modelů
  - porovnání výsledků simulací s požadavky
  - provázání požadavků, modelů a testů



https://www.mathworks.com/solutions/verification-validation.html

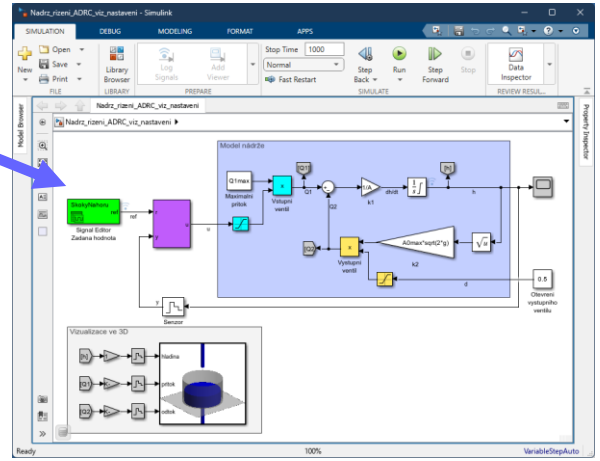
Ukázka: Testování modelu

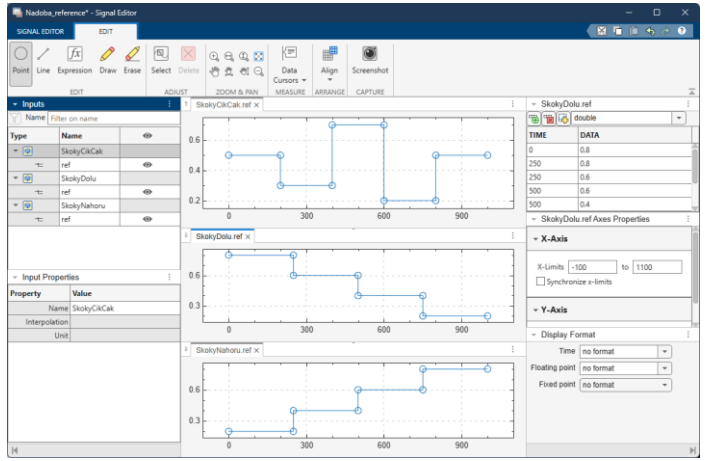
- Testování modelu pomocí simulace
- Manuální
  - změna žádané hodnoty a velikosti výtoku interaktivně
- Automatizované
  - jako posloupnost hodnot $ref \in \{0.2, 0.4, 0.6, 0.8\}$



manuální testování

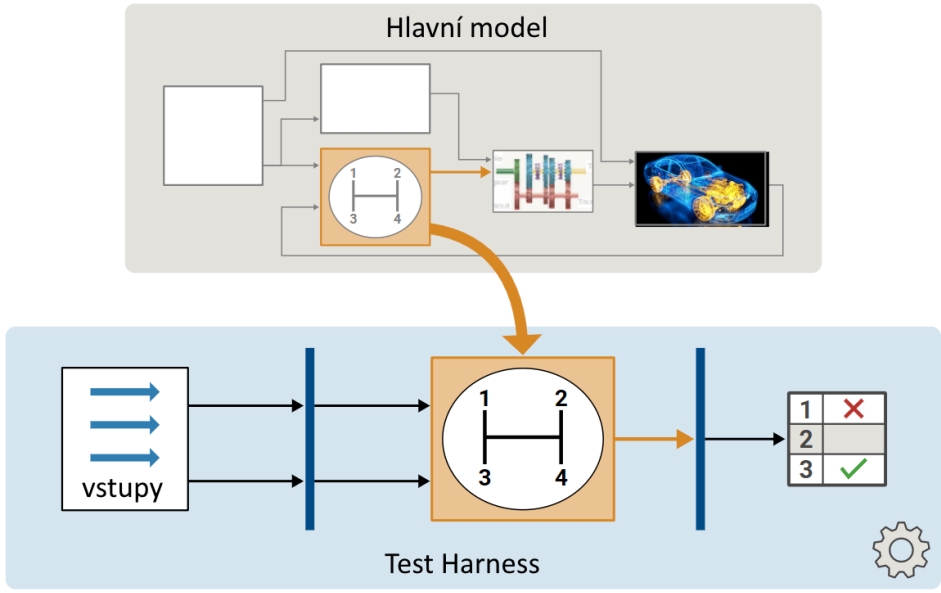
automatizované testování





Test Harness a Test Manager

- Test Harness = speciální testovací model
  - uložený s testovaným modelem nebo externě
  - lze aplikovat na celý model, vybraný subsystém v modelu, atd.

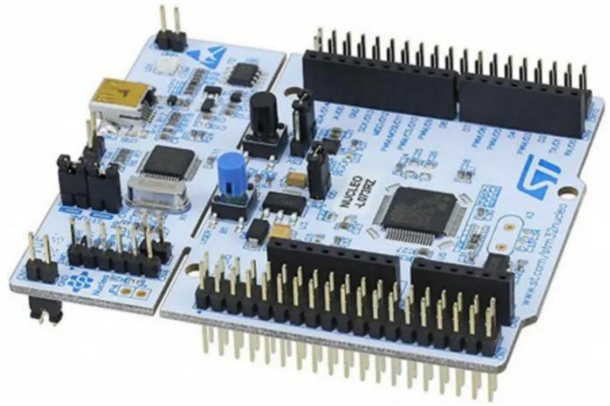


- Test Manager
  - grafické rozhraní pro tvorbu, správu, spouštění a hodnocení výsledků testů

https://www.mathworks.com/products/simulink-test.html

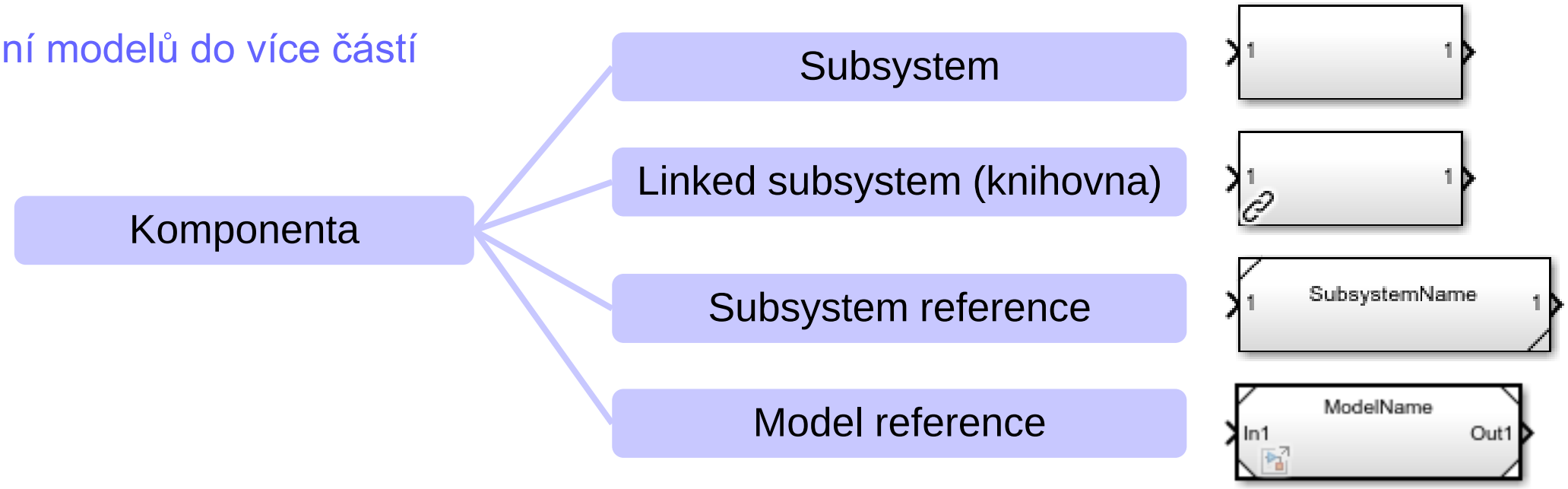
Testování v reálném čase

- Rapid Control Prototyping (RCP)
  - připojení modelu algoritmu v Simulinku k reálnému zařízení
  - ověření chování navrženého systému s reálnou soustavou
  - ladění parametrů
- Jaký použít hardware?
- Real-time simulátor ~ tradiční RCP
  - cíl: ověřit funkčnost algoritmu v reálném prostředí
  - výkon, flexibilita
- Vývojová embedded platforma ~ on-target RCP
  - cíl: ověřit praktickou realizovatelnost algoritmu pro výrobu



Integrace modelů komponent

- Komponentizace
  - členění modelů do více částí

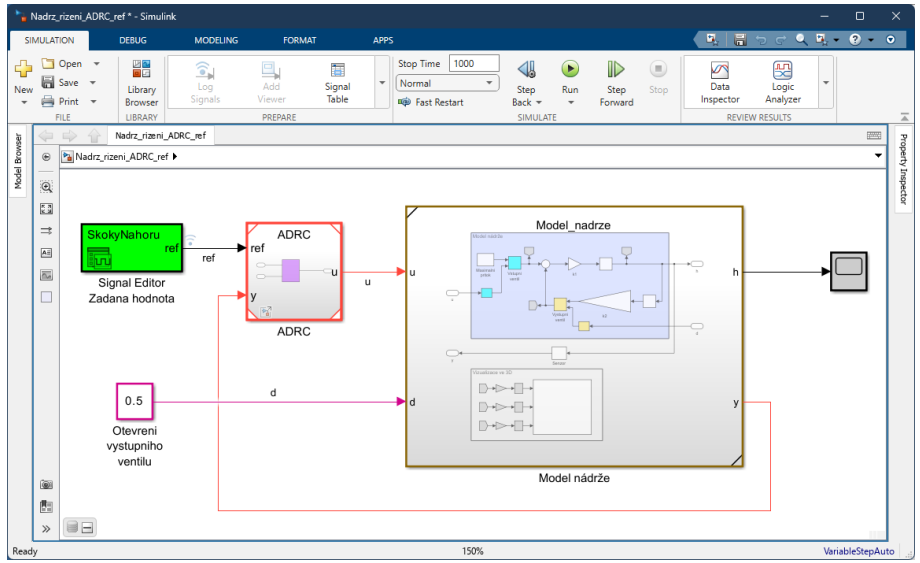


- Integrace modelů komponent
  - zapojení modelů komponent do modelu systému
  - propojení modelů až do kořenové úrovně nebo do modelu architektury

https://www.mathworks.com/help/simulink/ug/types-of-model-components.html

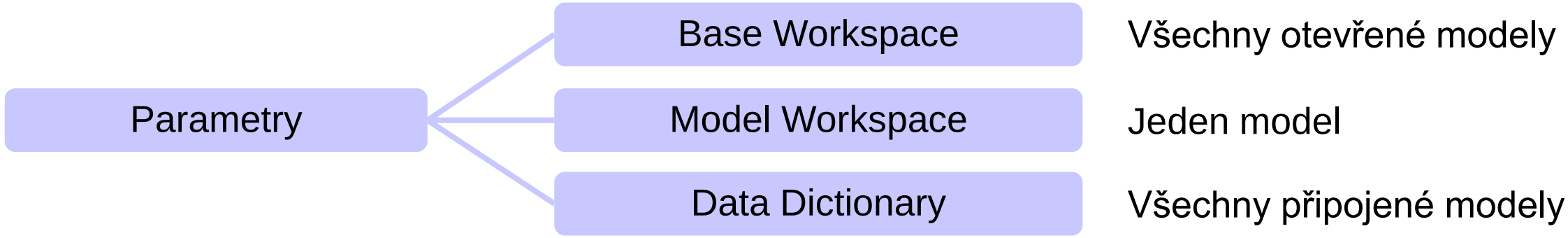
Ukázka: Rozdělení modelu systému do komponent

- Části modelu
  - řídicí systém ADRC
  - soustava
  - vstupní signály
- Typy komponent v modelu
  - ADRC = referencovaný model
    - když vyžadujeme možnost samostatného nastavení a simulace
  - Soustava = referencovaný subsystém
    - když nevyžadujeme možnost samostatného nastavení a simulace, ale chceme často upravovat
  - Alternativa: Soustava = knihovna
    - nevyžadujeme možnost samostatného nastavení a simulace
    - pro opakovaně využitelnou funkčnost menší komponenty, která se zřídka upravuje



Správa dat v modelech

- Kde mohou být umístěny parametry

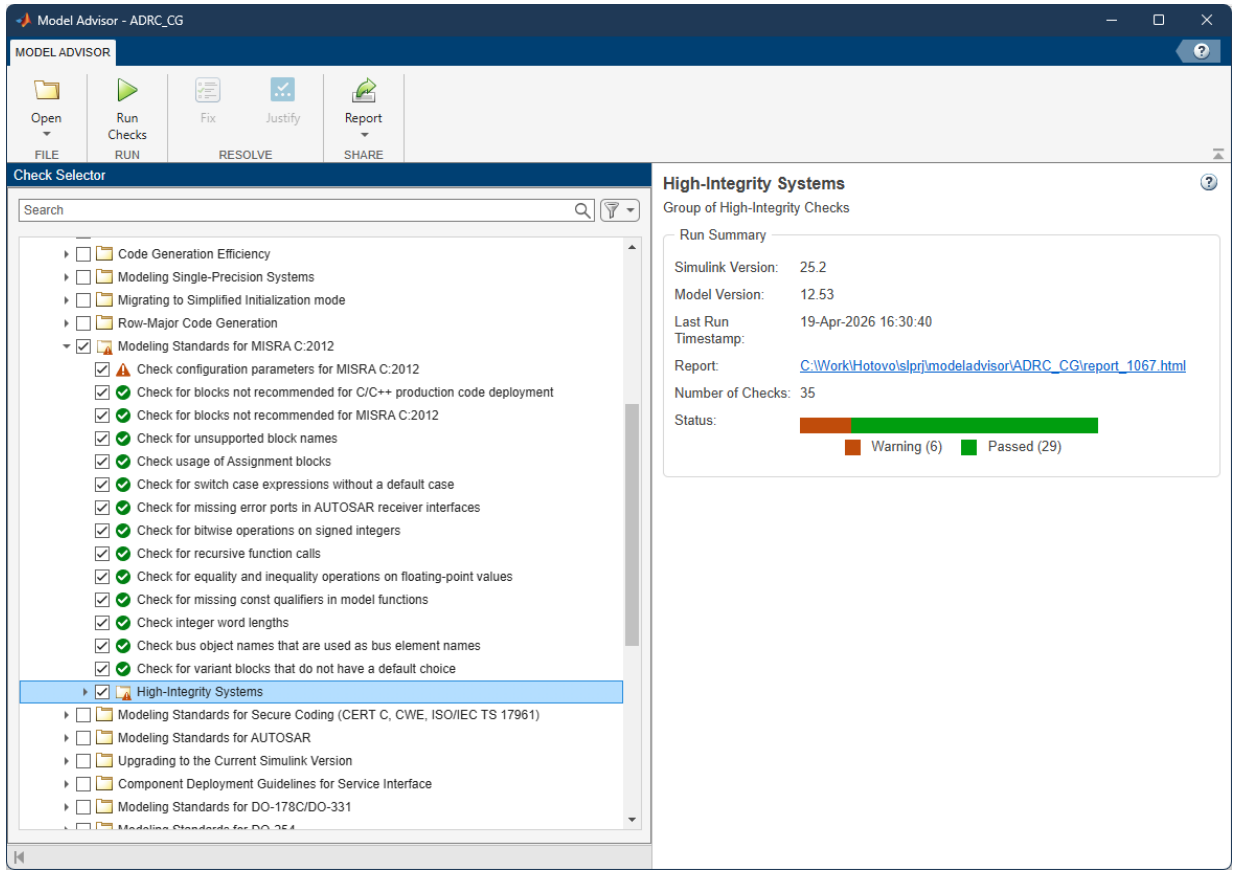


- Jak mohou být uspořádány signály

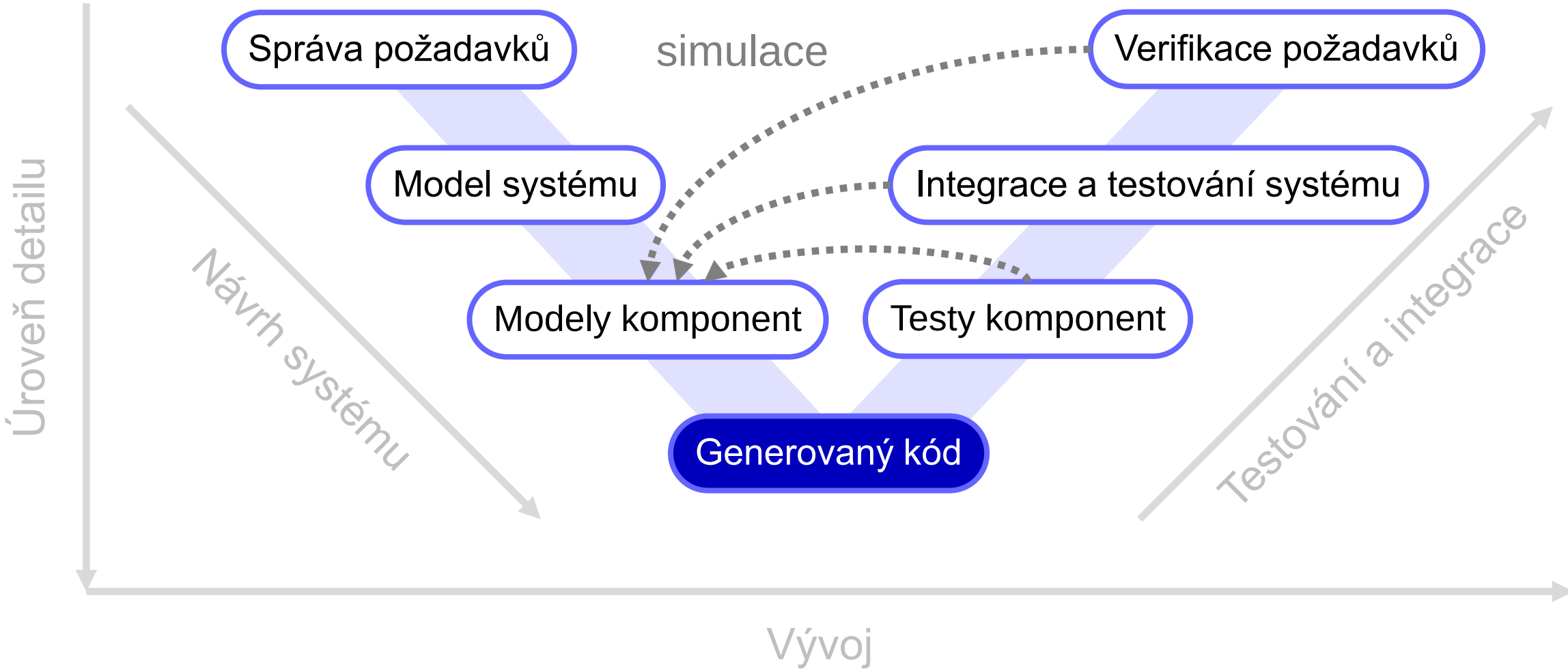
https://www.mathworks.com/help/simulink/ug/determine-where-to-store-data-for-simulink-models.html

Ukázka: Kontrola souladu se standardy

- Kontrola souladu ADRC se standardem MISRA-C

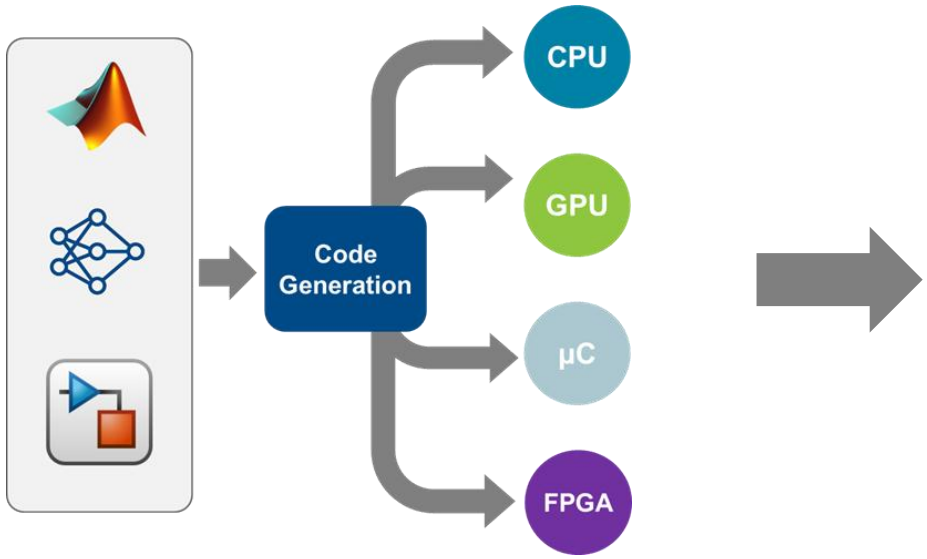


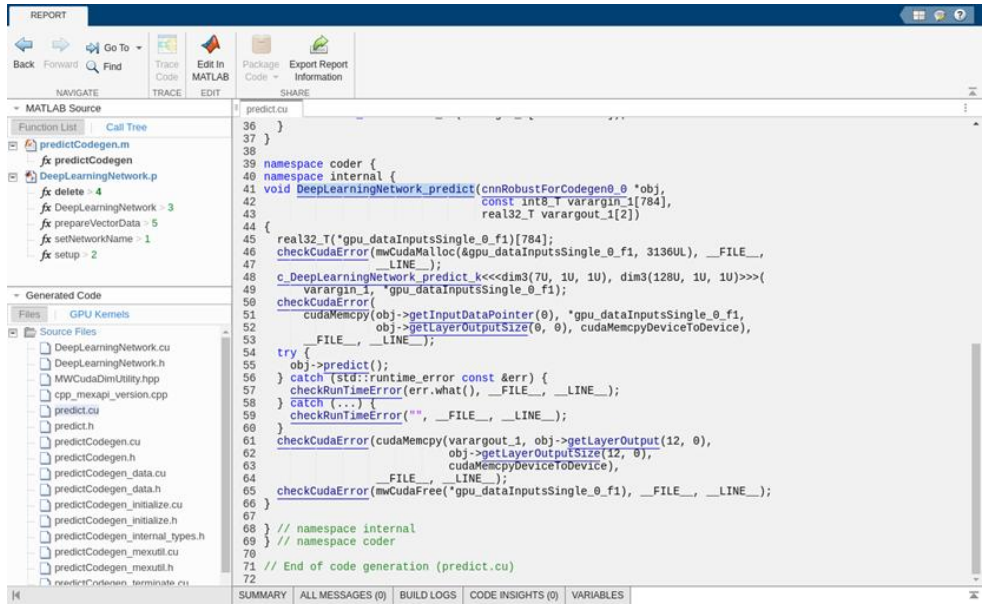
Automatické generování kódu



Implementace modelu

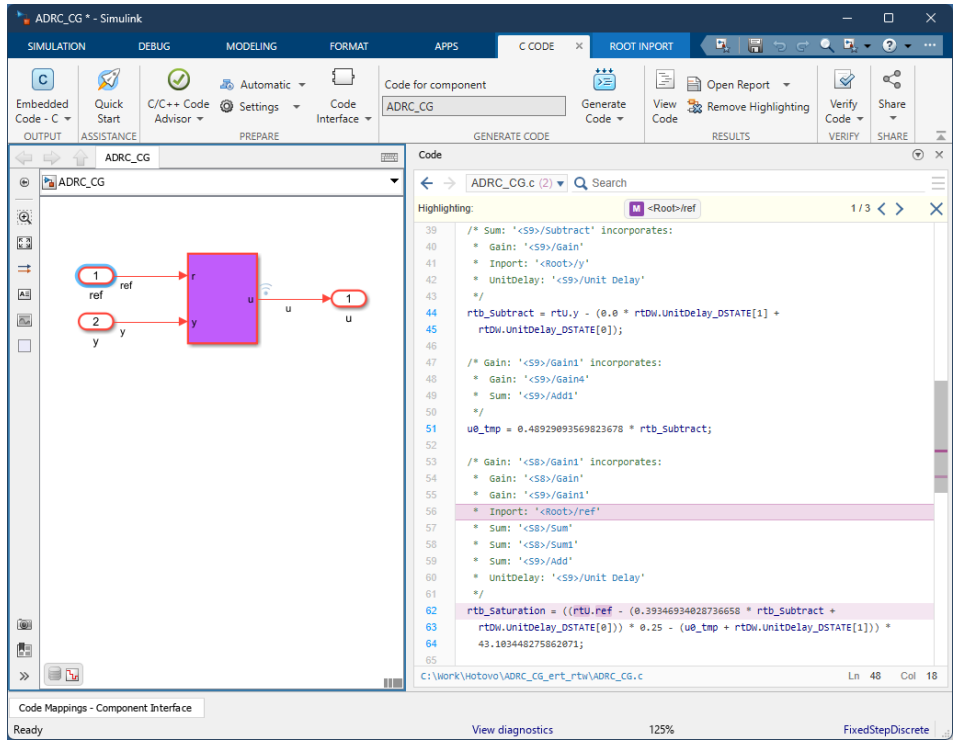
- Příprava modelu pro implementaci na cílovou platformu
  - Embedded Coder Quick Start
  - nastavení vzhledu generovaného kódu a optimalizace pro zvolenou cílovou platformu
- Automatické generování kódu



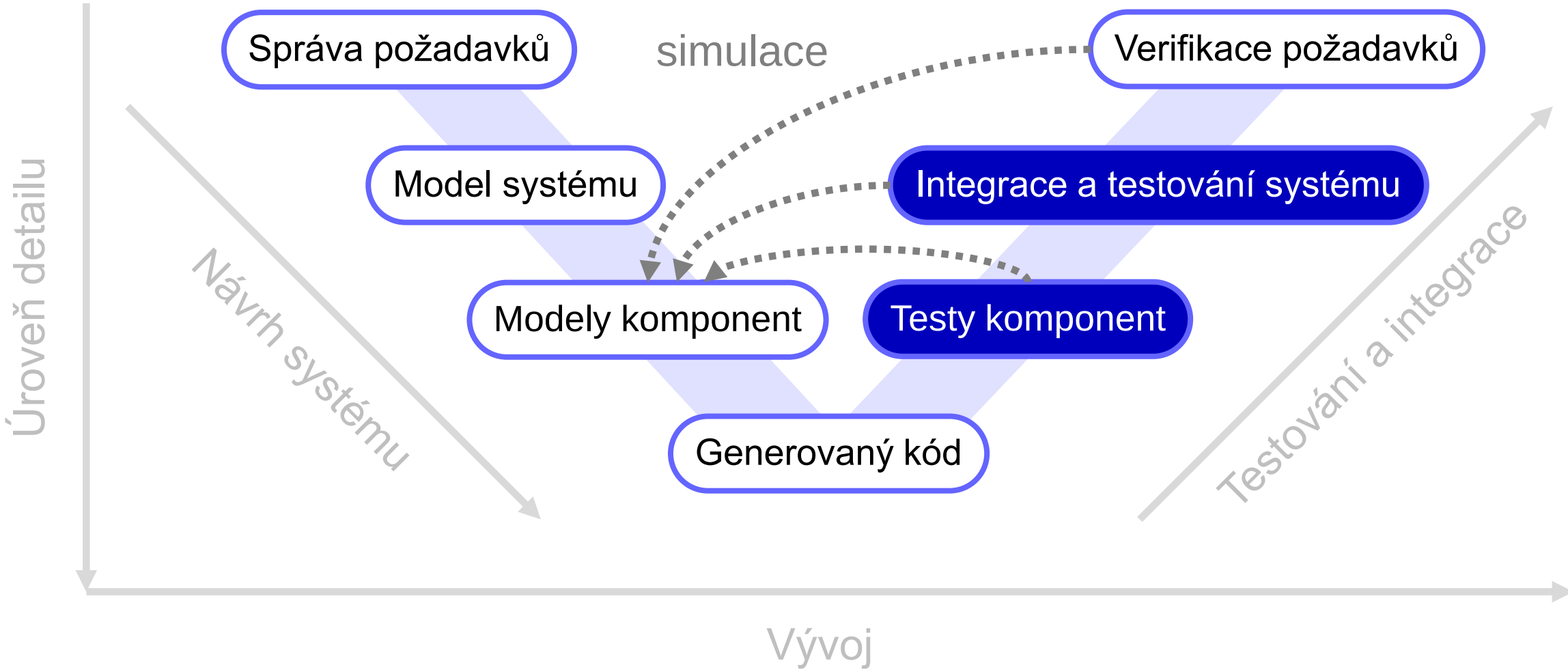


Ukázka: Generování kódu

- Model řídicího systému
  - bez modelu soustavy a testovacích vstupů
- Embedded Coder
  - nastavení parametrů pro generování kódu
- Trasování z modelu do kódu a opačně



Testování a integrace kódu

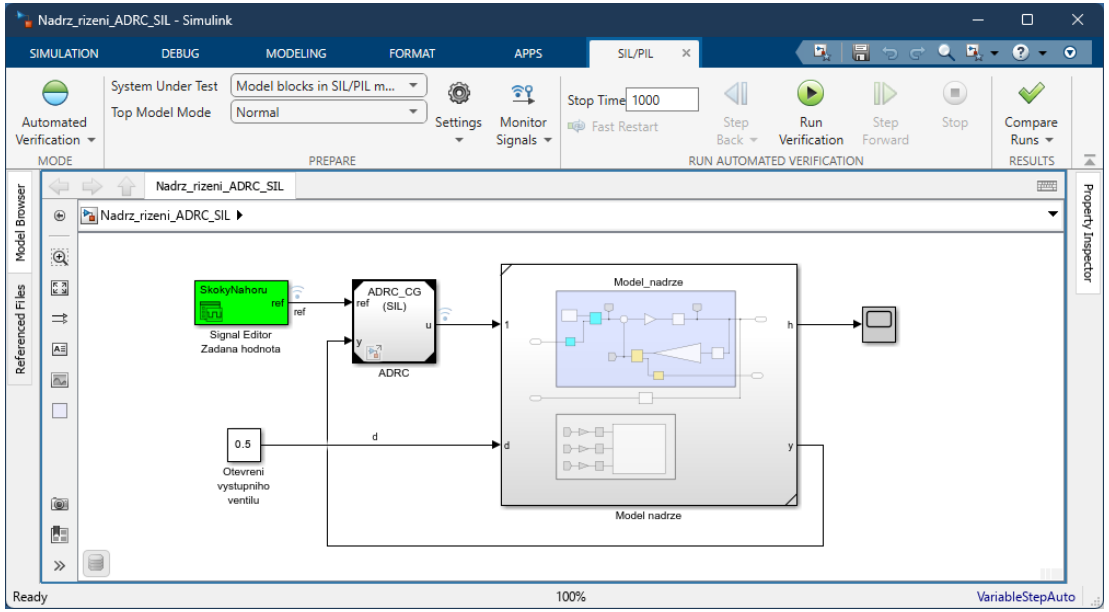


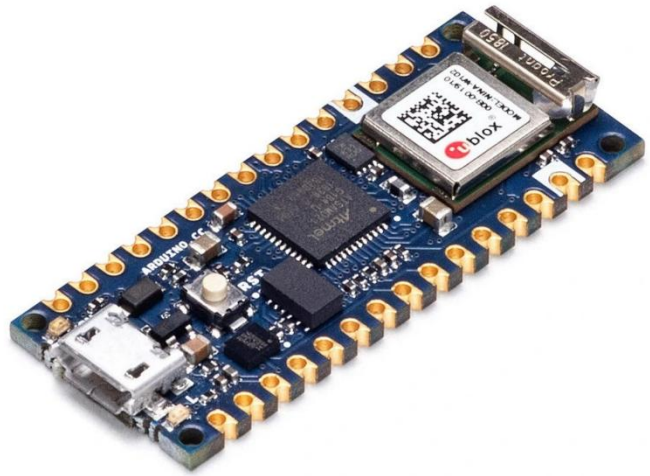
Testování a integrace kódu

- Testování implementovaného algoritmu
  - ověření chování algoritmu v implementované podobě
  - zavedení generovaného kódu zpět do Simulinku a provedení testů: *software-in-the-loop* (SIL)
  - propojení Simulinku s cílovou platformou, testování na cílovém HW: *processor-in-the-loop* (PIL)
- Integrace kódu
  - zapojení generovaného kódu do širšího rámce
  - propojení s externími částmi kódu
- Simulink jako platforma pro integraci
  - integrace může být řešena současně s předchozí fází implementace
  - bloky v prostředí Simulink pro integraci C/C++ kódu
  - výsledný kód obsahuje generované části společně s integrovanými

Ukázka: SIL a PIL simulace

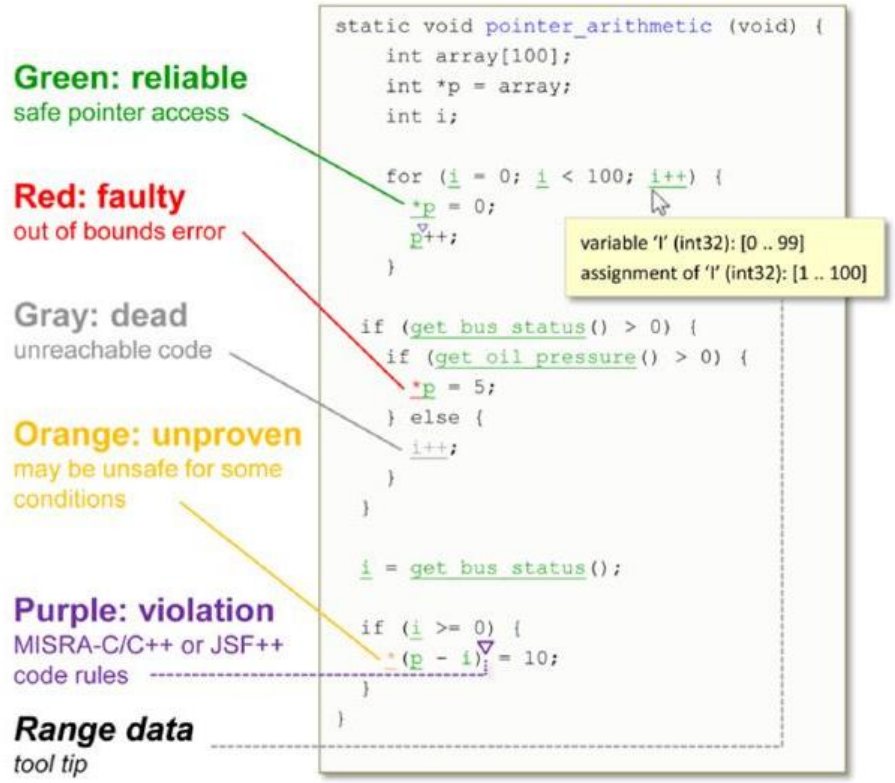
- Model řídicího systému v implementované podobě
- Provedení simulačních testů s generovaným kódem
  - software-in-the-loop
  - processor-in-the-loop (Arduino Nano 33 IoT)





Verifikace kódu

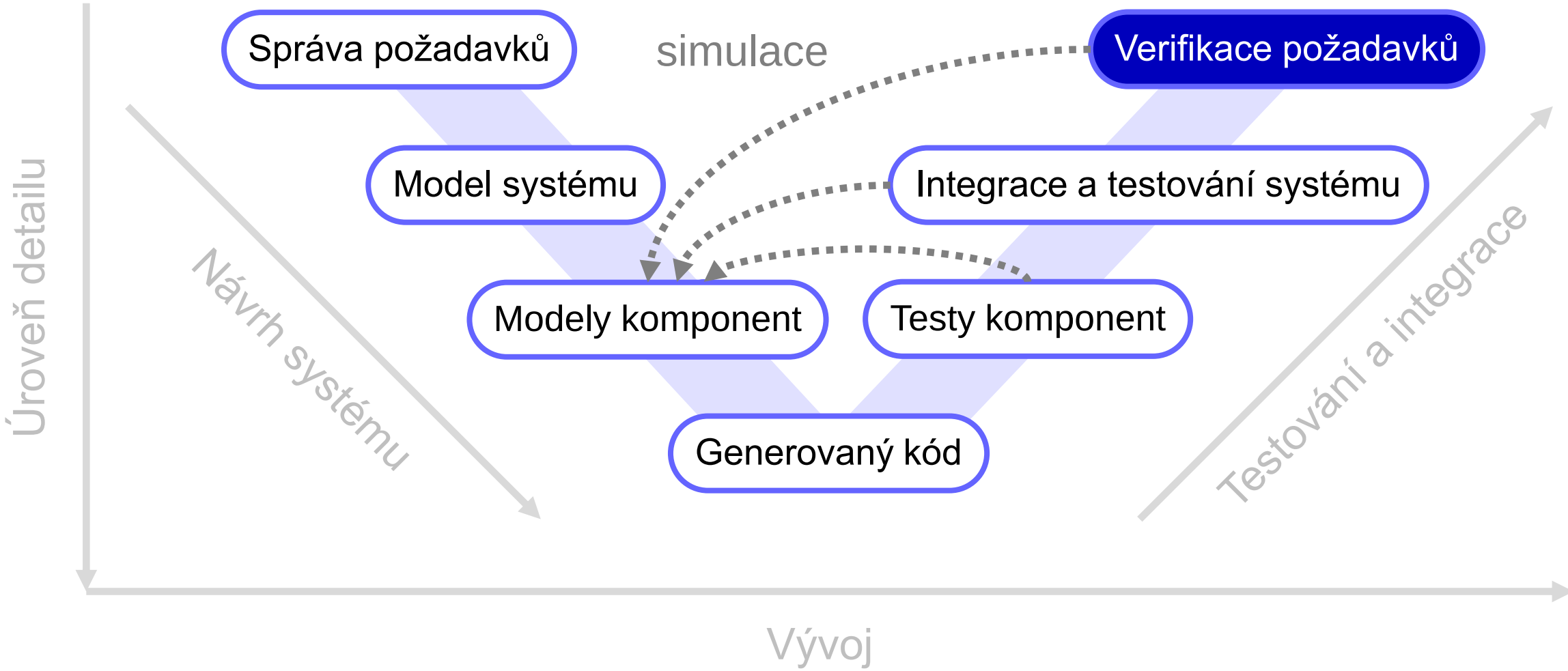
- Verifikace implementovaného algoritmu
  - zásadní zejména pro bezpečnostně kritické aplikace
- Formální analýza kódu
  - nástroje Polyspace
  - prokázaní nepřítomnosti run-time chyb
  - využívá formální metody (= bez spouštění)



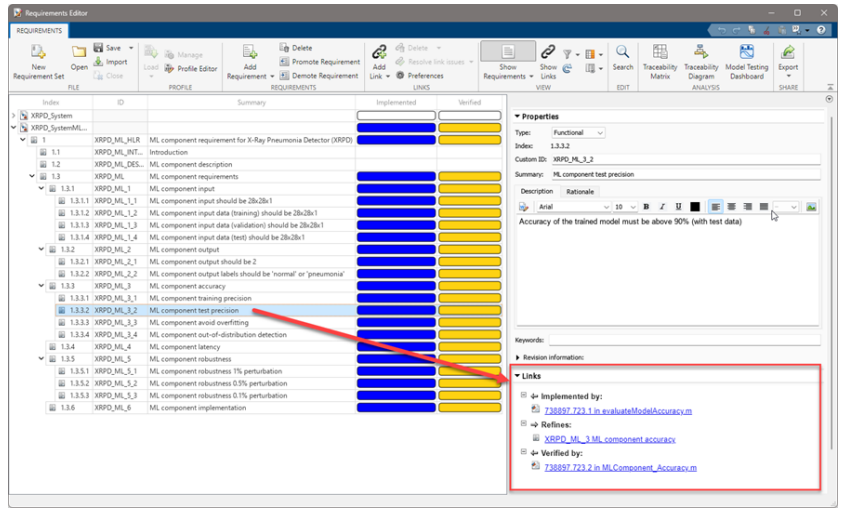
Testování vyrobeného zařízení

- Testování v reálném čase
  - *hardware-in-the-loop* (HIL)
- Typické uspořádání
  - vyvinuté reálné zařízení připojeno k simulačnímu modelu soustavy
  - simulační model spuštěn v reálném čase
  - ~ chová se jako reálný systém
- Real-Time simulátor
  - periférie odpovídají připojení reálného systému
  - ~ připojuje se jako reálný systém

Verifikace požadavků



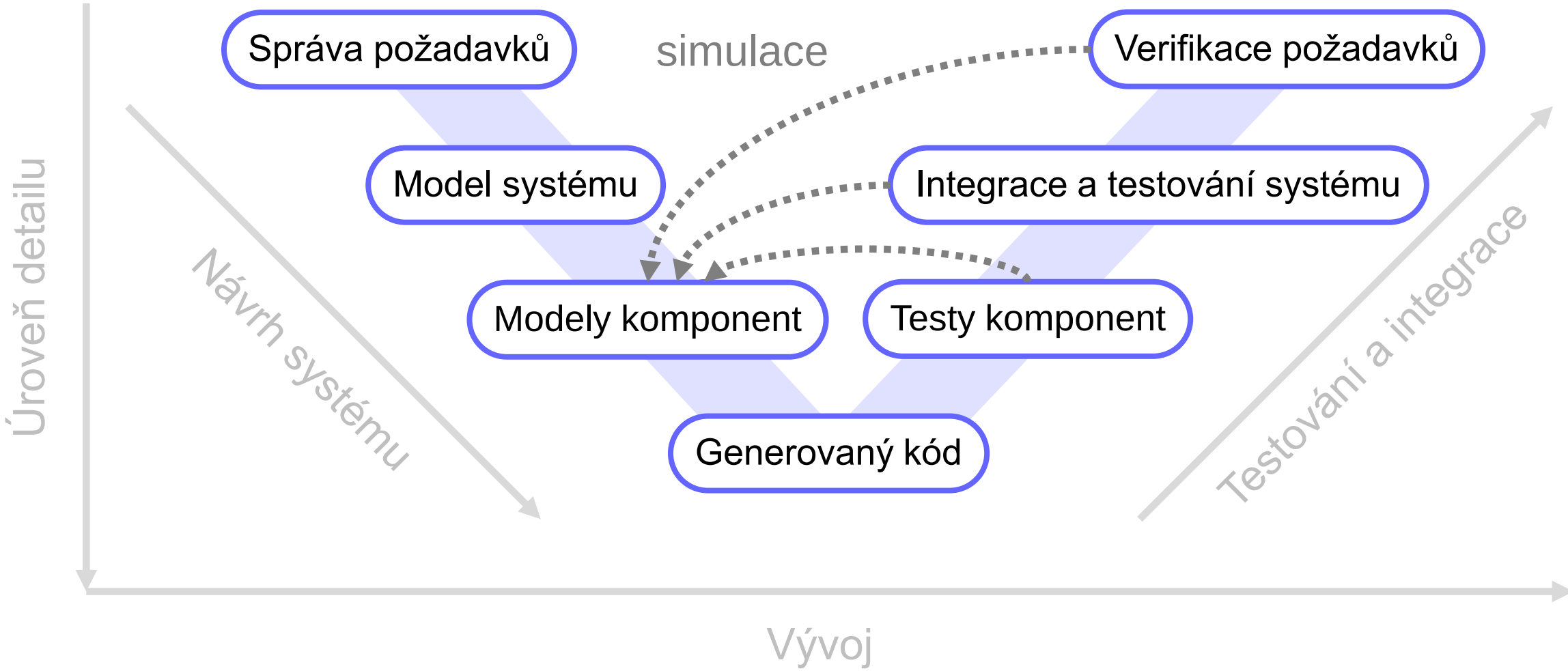
Verifikace požadavků



- Závěrečná část V-cyklu
- Ověření, že byly veškeré požadavky
  - a) implementovány v podobě navrhované funkčnosti
  - b) splněny v podobě provedených testů
- Provázání požadavků s implementovanými funkcemi a testy
  - klíčové pro uzavření smyčky vývojového procesu
  - spuštěním všech definovaných testů ověříme, že požadavky byly adekvátně implementovány

Index	ID	Summary	Implemented	Verified
XRPD_System				
XRPD_SystemMLC...				
1	XRPD_ML_HLR	ML component requirement for X-Ray Pneumonia Detector (XRPD)		
1.1	XRPD_ML_INT...	Introduction		
1.2	XRPD_ML_DES...	ML component description		
1.3	XRPD_ML	ML component requirements		
1.3.1	XRPD_ML_1	ML component input		
1.3.1.1	XRPD_ML_1_1	ML component input should be 28x28x1		
1.3.1.2	XRPD_ML_1_2	ML component input data (training) should be 28x28x1		

Shrnutí: Vývojový V-cyklus pro (řídicí) systémy s využitím MBD



Mapa nasazení metody MBD

	Modelování a simulace	*Simulace a testování v reálném čase*	*Vývoj produkčního kódu*
Systematické testování algoritmů	Verifikace systému	Automatizované testování v reálném čase	Certifikace
Simulace algoritmů s modelem systému	Simulace systému	Hardware-in-the-Loop (HIL) simulace	Software- a Processor-in-the-Loop (SIL, PIL) simulace
Vývoj algoritmů	Modelování algoritmů	Rapid Prototyping	Generování embedded kódu

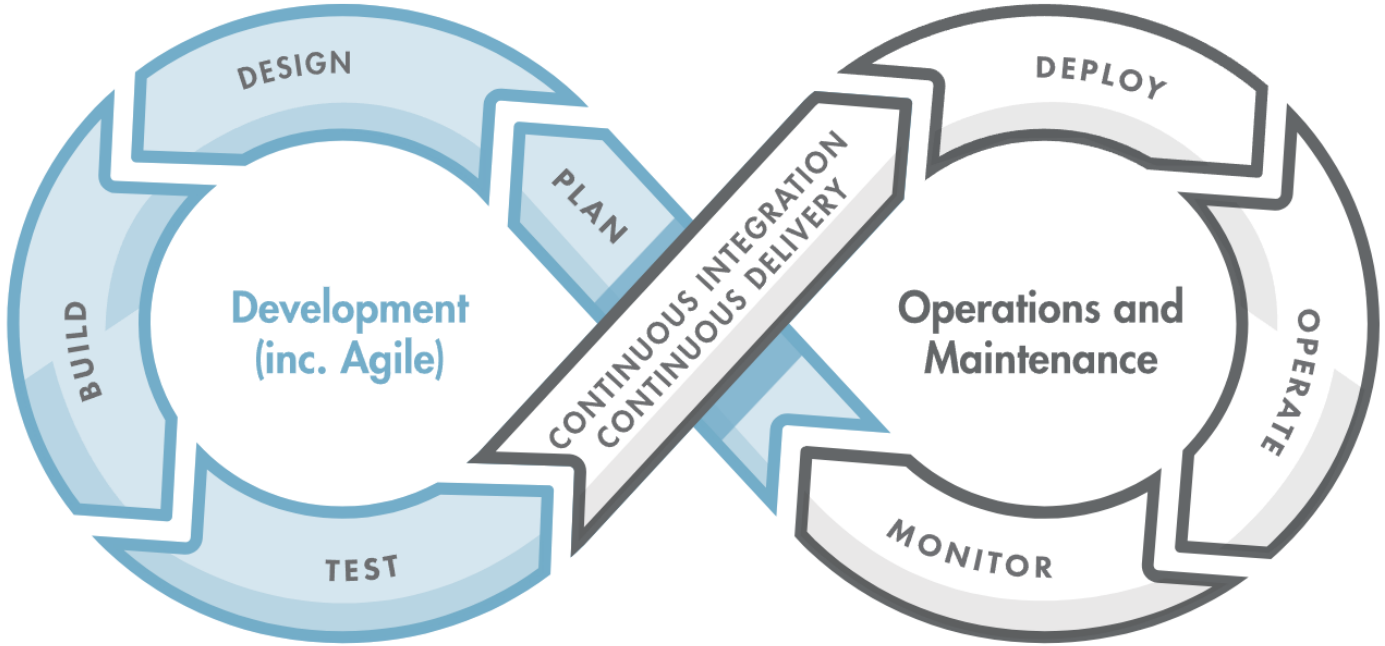
Používání modelů

Žádné generování kódu → Generování kódu pro testy → Generování kódu pro produkci

Generování kódu

DevOps

- Zajišťujte průběžné aktualizace softwaru
- Iterace mezi vývojem a provozem systému
- Využívá simulace, automatizované testování a generování kódu



Děkuji za pozornost.

Otázky?

Kontakty

HUMUSOFT®

- Pro technické dotazy využijte naši technickou podporu
  - email: support@humusoft.cz
  - tel.: +420 602 231 500
  - https://www.humusoft.cz/matlab/support/
- Všeobecné dotazy
  - email: info@humusoft.cz
  - tel.: +420 284 011 730
- Nebo můžete využít náš kontaktní formulář
  - https://www.humusoft.cz/contact/#contact

Adresa:
Pobřežní 20
186 00 Praha 8
Česká republika

Web:
www.humusoft.cz