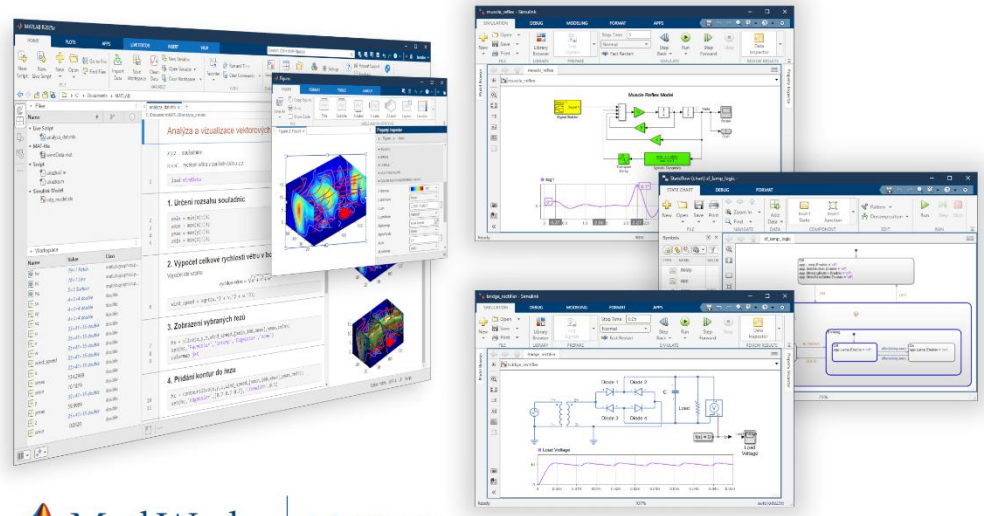
Embedded AI
s prostředím MATLAB a Simulink



MathWorks Authorized Reseller | MATLAB & SIMULINK

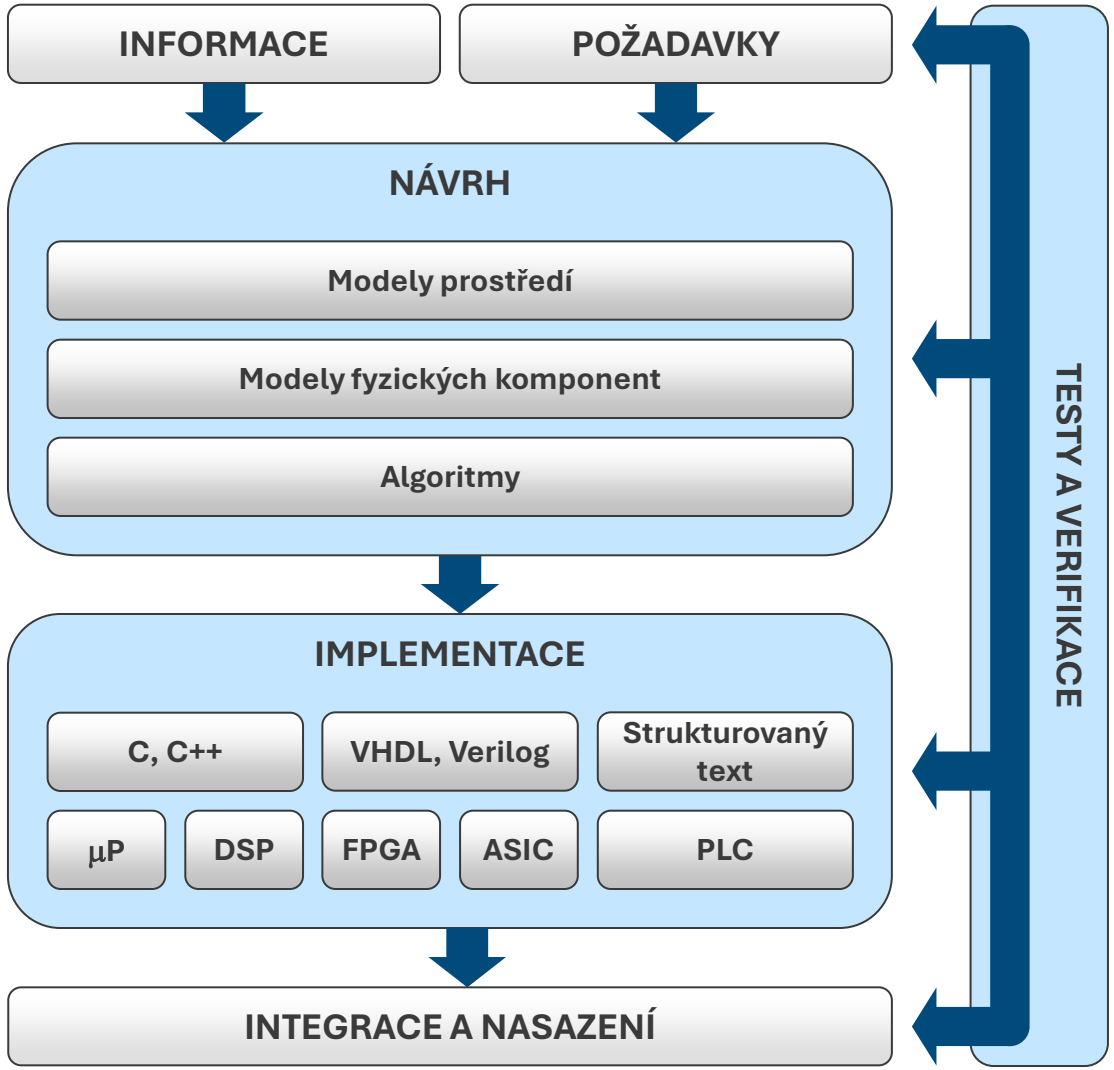
Jaroslav Jirkovský

jirkovsky@humusoft.cz

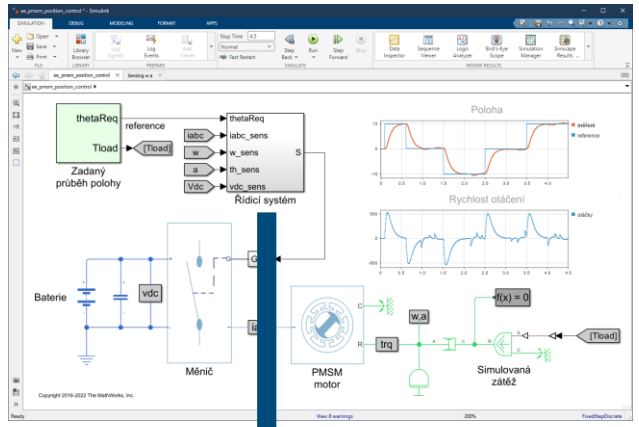
www.humusoft.cz
info@humusoft.cz

www.mathworks.com

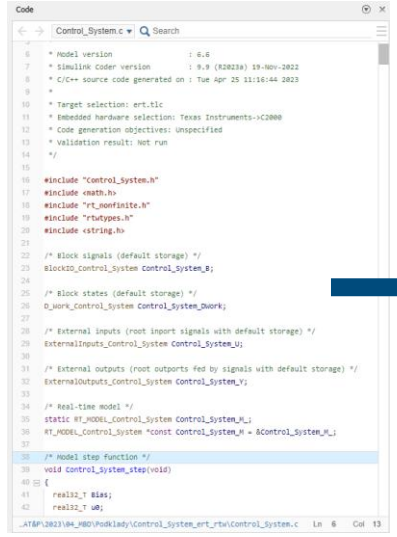
Vývoj metodou Model-Based Design



Modelování, simulace a testování



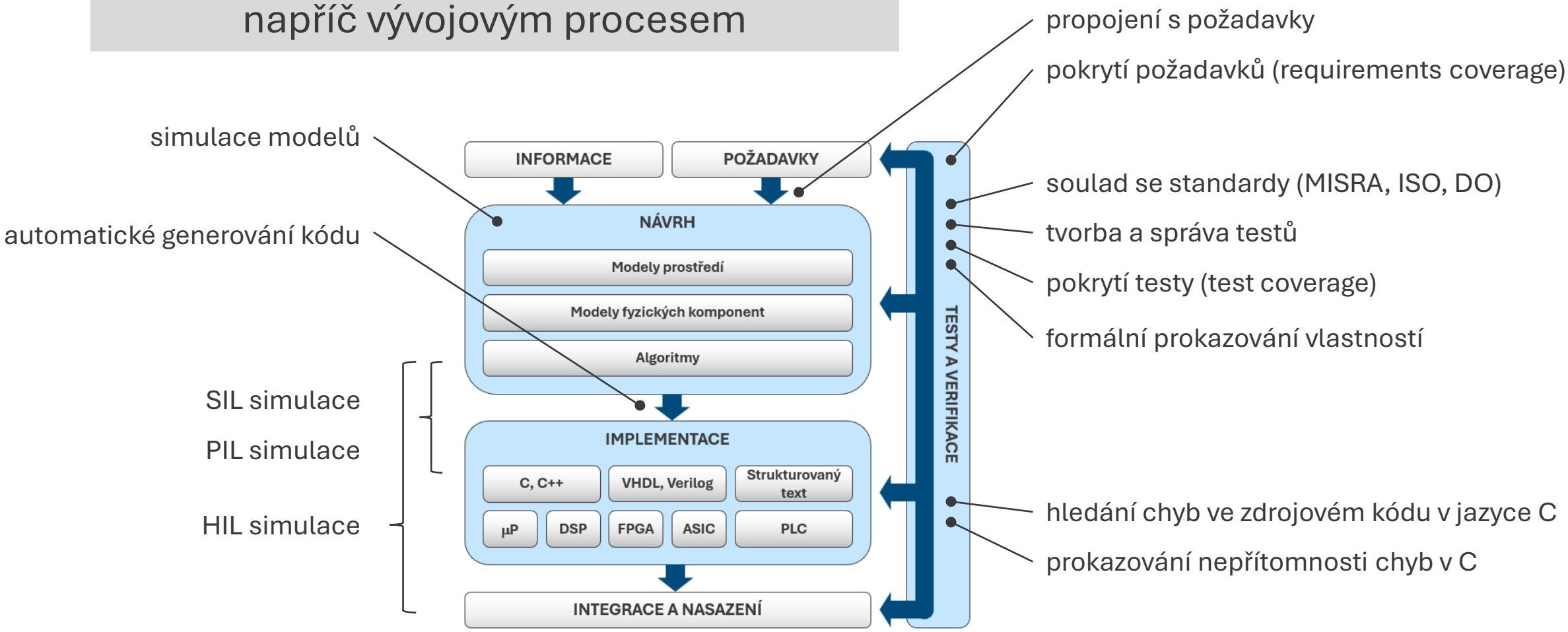
Automatické generování kódu



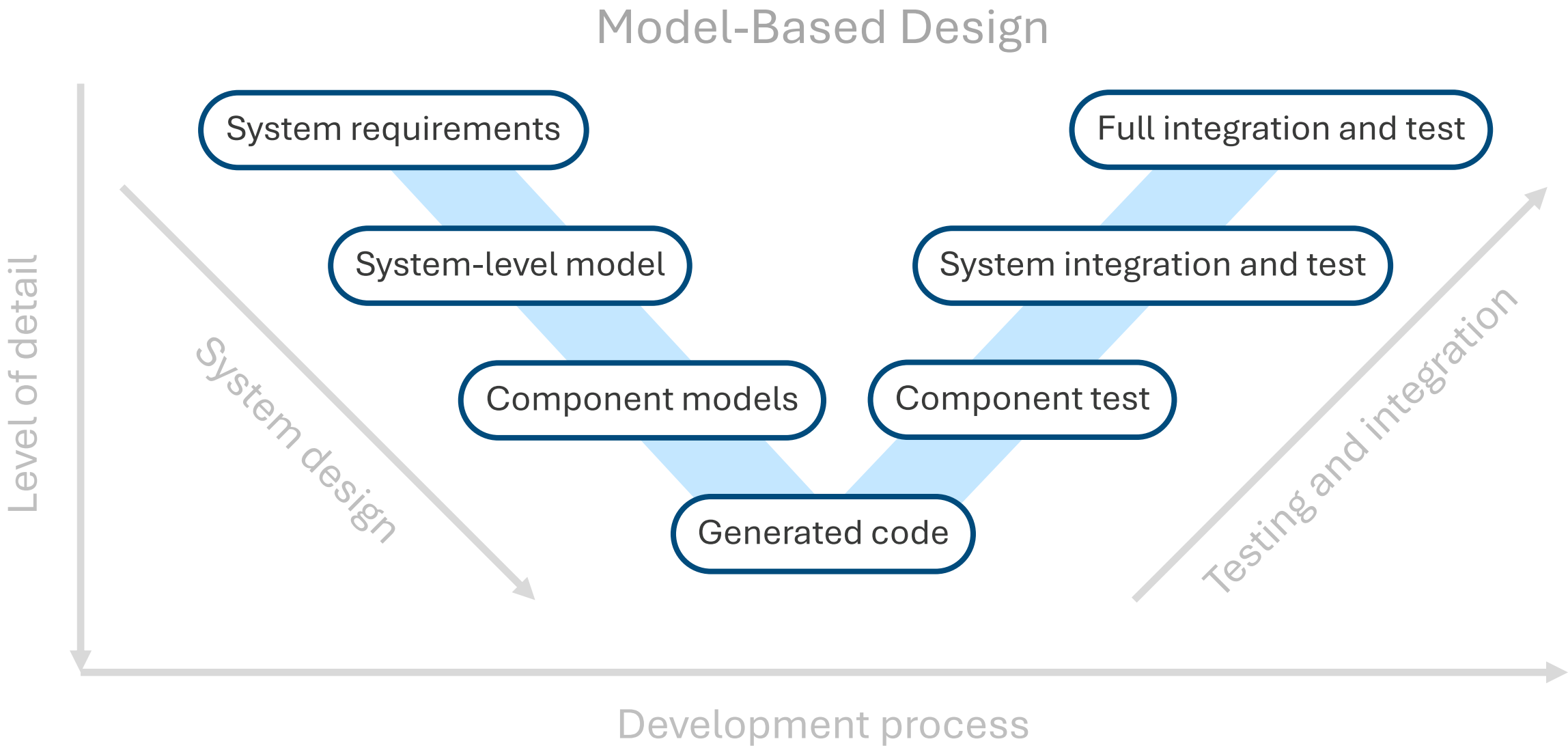
Nasazení

Vývoj metodou Model-Based Design

Systematické využití simulačních modelů napříč vývojovým procesem



Vývojový cyklus V pro standardní (ne-AI) embedded systémy



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

Typy AI

Diskriminativní AI

klasifikace a predikce výstupu na základě vstupních dat

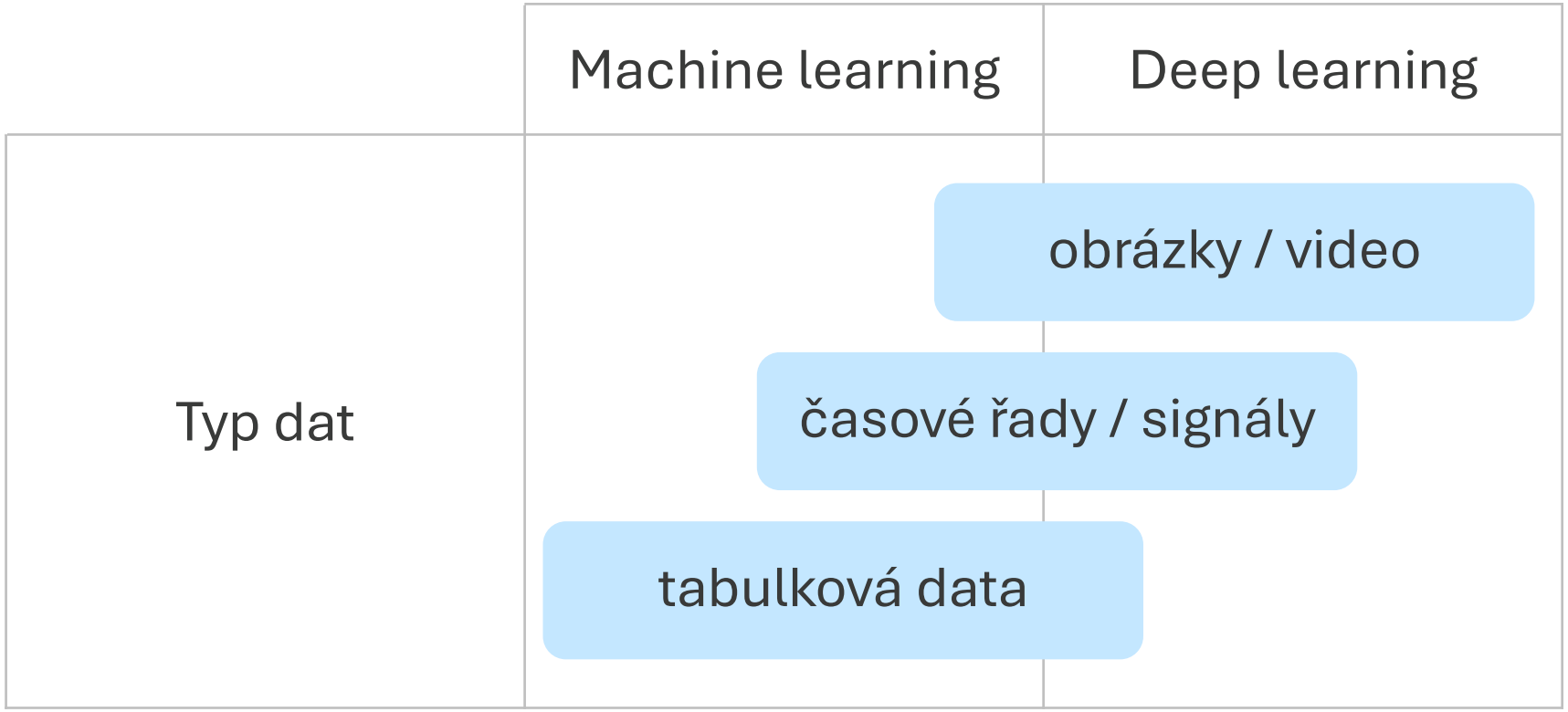
zaměření dnešního semináře

~~Generativní AI~~

AI Chat	generování a simulace nových dat podobných tomu, co se AI model naučil
AI Copilot	+ integrace a kooperace v rámci daného prostředí
Agentic AI	+ aktivní provádění akcí k dosažení zadaných cílů

https://www.mathworks.com/products/matlab/generative-ai.html

Diskriminativní AI ~ machine learning a deep learning



Cloud AI, Edge AI, Embedded AI a tinyML

Cloud AI

AI modely, které jsou spouštěny na výkonných serverech ve vzdálených datových centrech. Ideální pro učení komplexních modelů a spouštění LLM.

Edge AI

Provozování AI modelu se odehrává na lokálním hardware, v blízkosti zdroje dat.

Lokální servery

Průmyslová PC

Embedded AI

AI modely které jsou spouštěny na zařízeních s omezenými zdroji jako součást vyvíjeného systému

Embedded GPU

Mikrokontroléry

NPU

tinyML

FPGA

Mikroprocesory

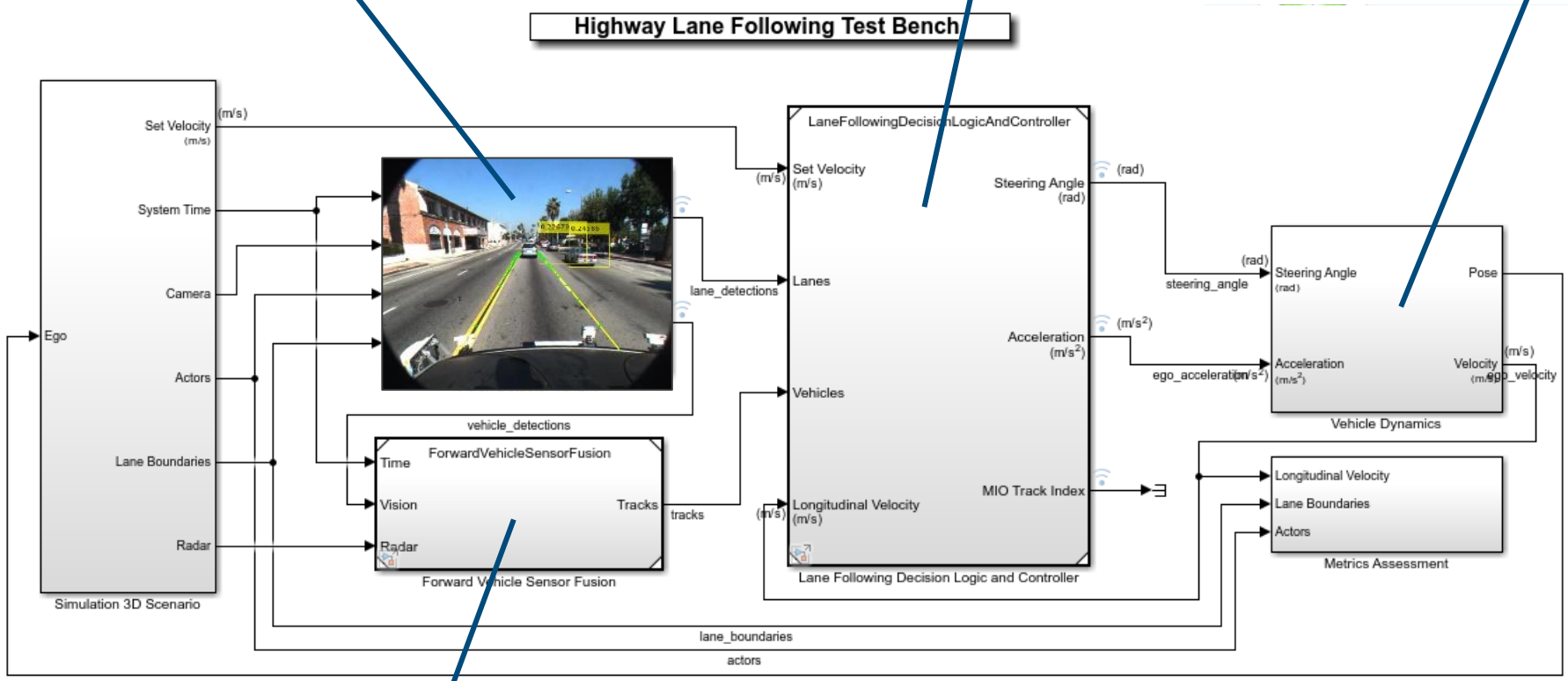
Latence, zpracování, energie

Typický příklad: Asistenční systém automobilu

AI algoritmus
detekce jízdních pruhů a vozidel

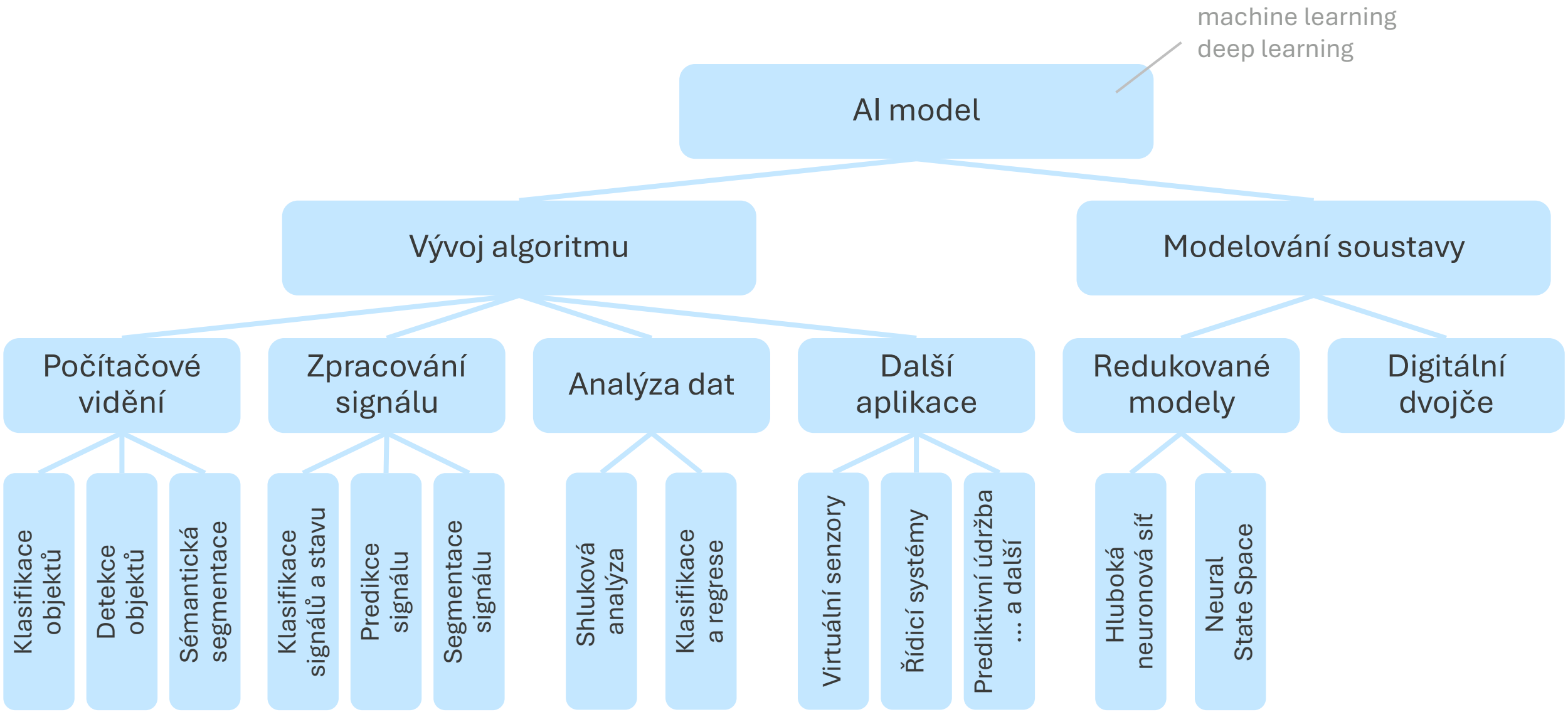
Řídicí systém

Model dynamiky vozidla
pro testování chování systému



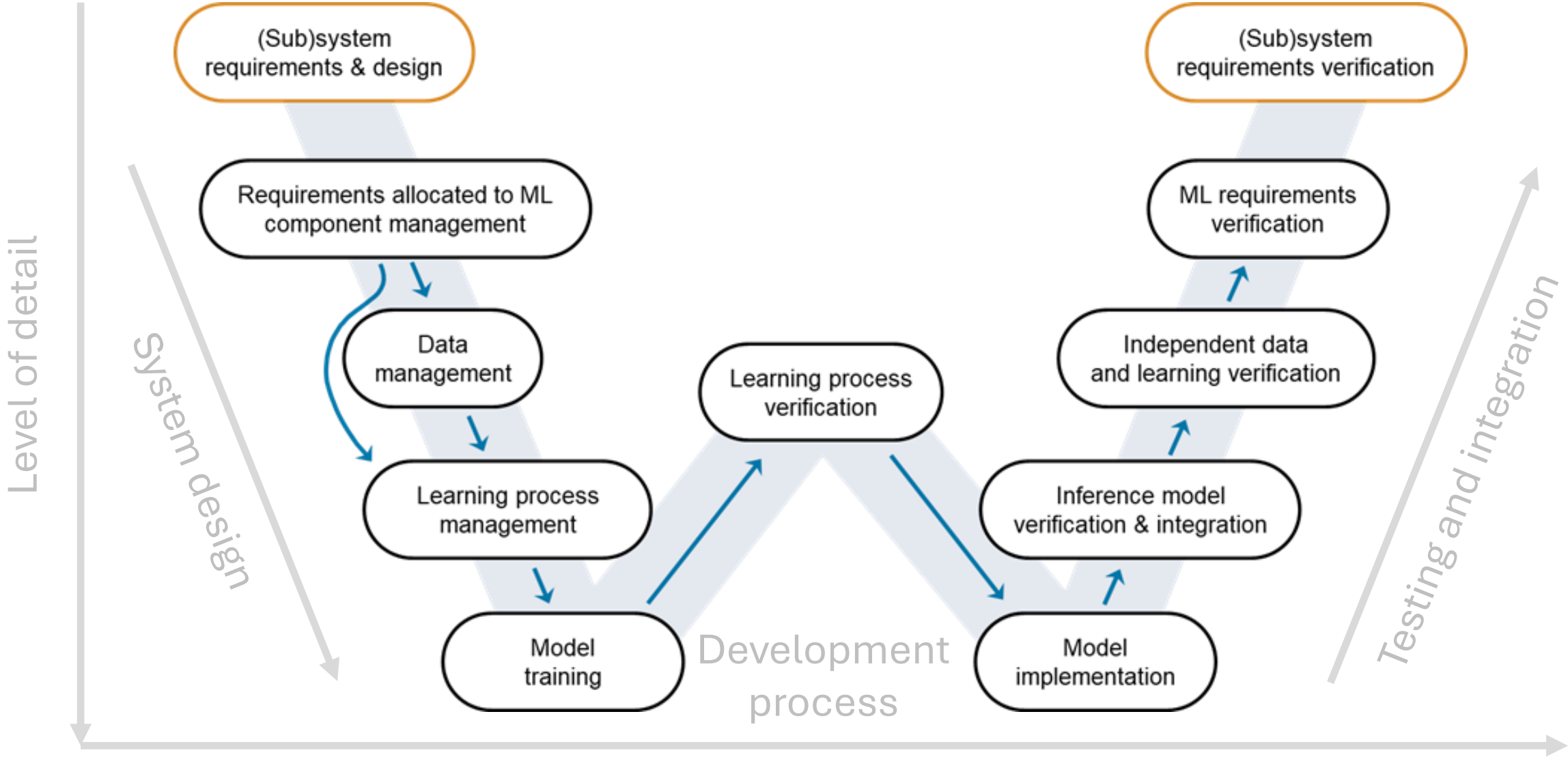
Senzorická fúze

AI modely a jejich využití



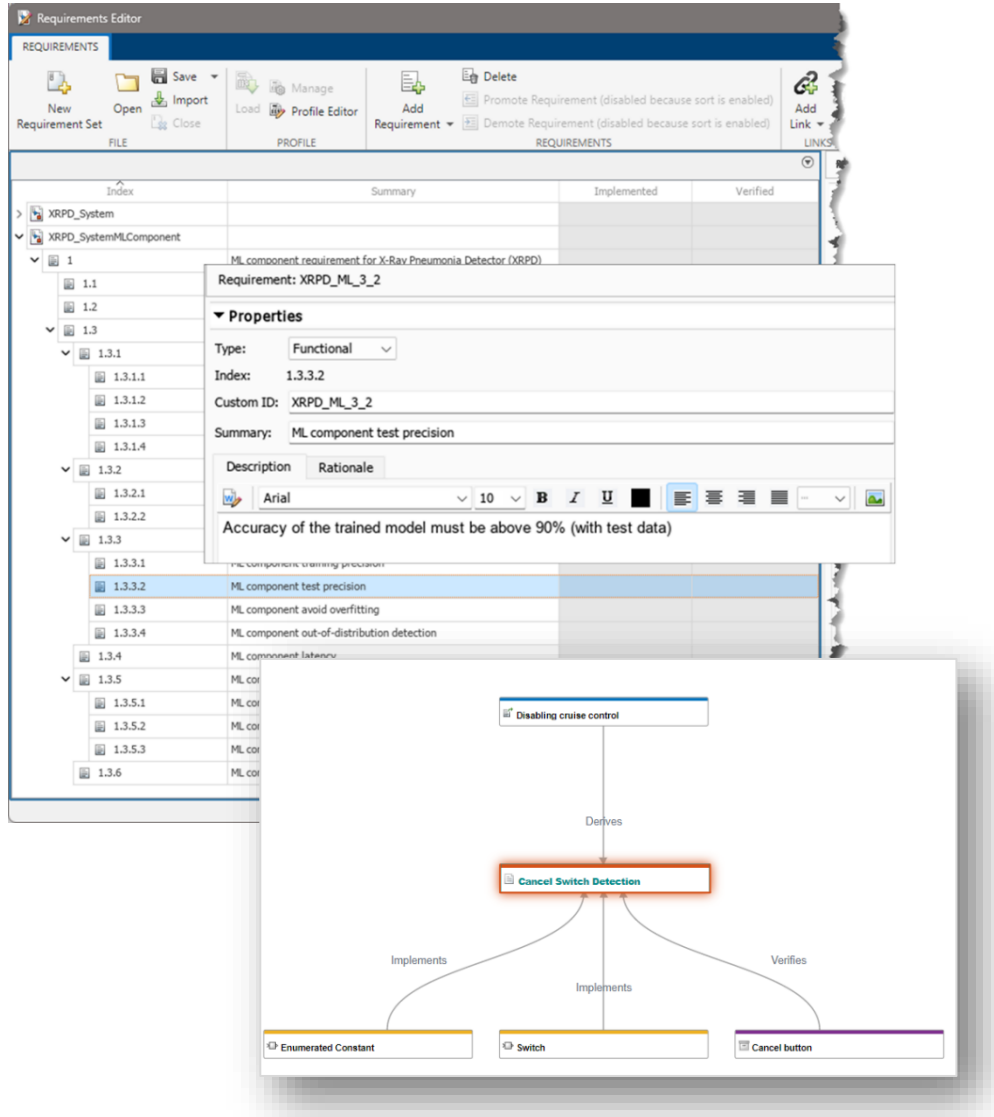
Vývojový cyklus W: Adaptace V-cyklu pro AI

Credit: EASA, Daedalean.

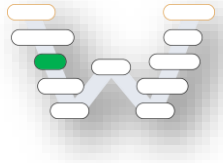


Správa požadavků

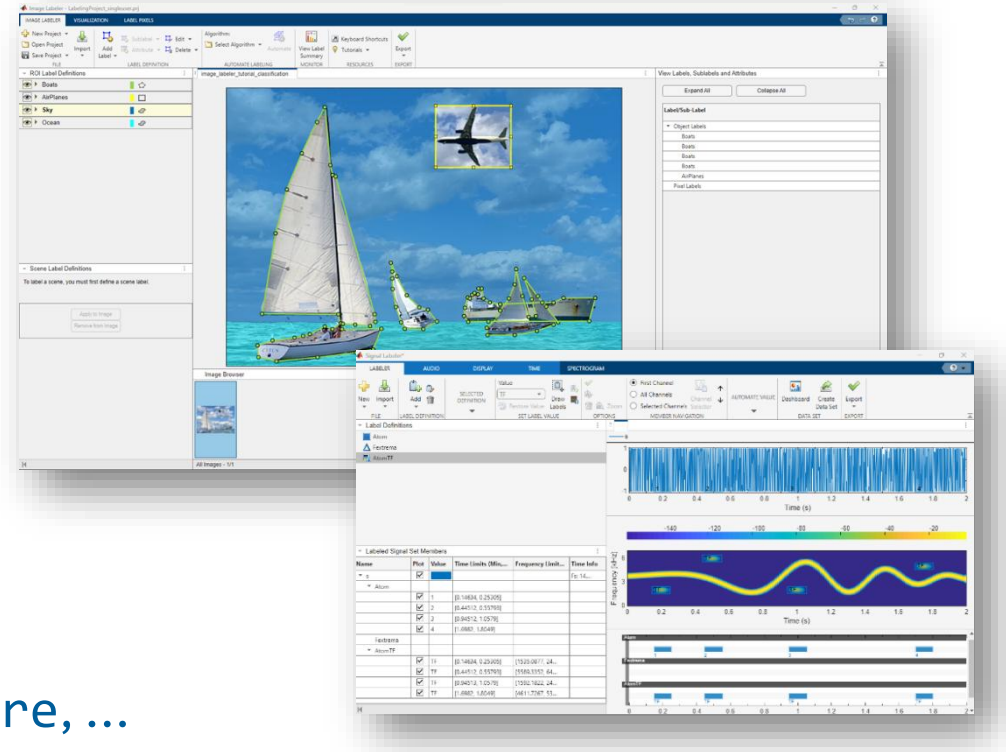
- Požadavky na celkový návrh systému
- Požadavky specifické pro AI komponenty
- Requirements Toolbox
  - vytváření, propojení a trasování požadavků
  - Requirements Editor
  - Traceability Matrix
  - Traceability Diagrams
- Klíčové otázky
  - jsou všechny požadavky implementovány?
  - jak budou požadavky testovány?
  - lze vysvětlit chování modelu?



Správa dat

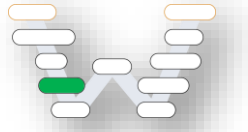


- Správa dat určených pro učení AI modelu
- Označování dat pro účely učení
  - Image Labeler, Signal Labeler
- Manipulace s daty
  - přímé načítání dat – pro malé datové sady
  - objekty datastore: `imageDatastore`, `signalDatastore`, ...
  - úprava dat: `TransformedDatastore`, `augmentedImageDatastore`, ...
- Rozsáhlá data
  - tall arrays

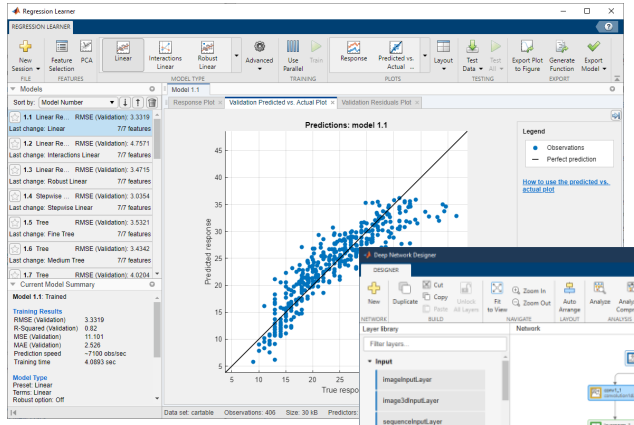


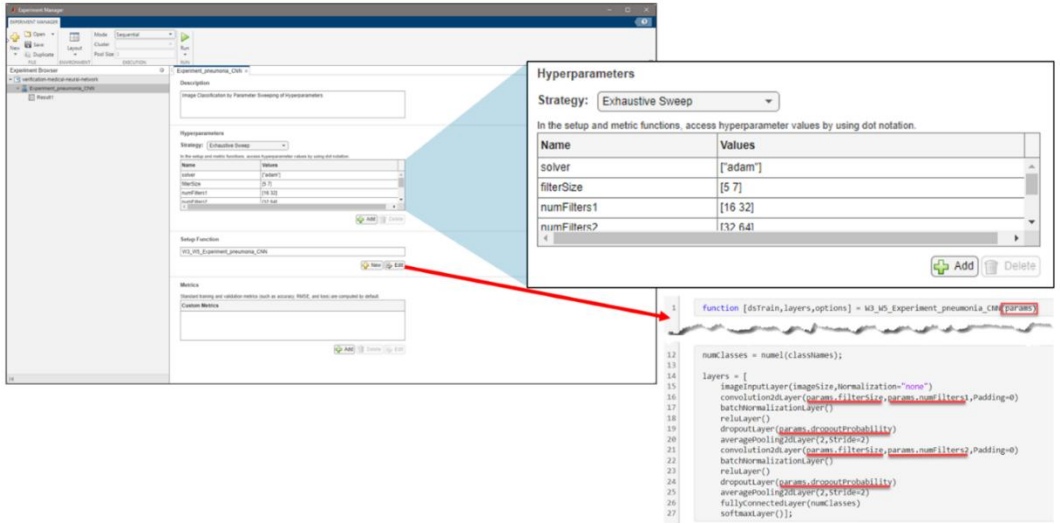
```
imds = ImageDatastore with properties:
                      Files: {
                             '...\matlab\toolbox\matlab\demos\example.tif';
                             '...\matlab\toolbox\matlab\matlab_images\tif\corn.tif'
                             }
                     Folders: {
                             '...\matlab\toolbox\matlab'
                             }
                      Labels: [demos; imagesci]
    AlternateFileSystemRoots: {}
                    ReadSize: 1
      SupportedOutputFormats: ["png"    "jpg"    "jpeg"    "tif"    "tiff"]
           DefaultOutputFormat: "png"
                     ReadFcn: @readDatastoreImage
```

Správa procesu učení



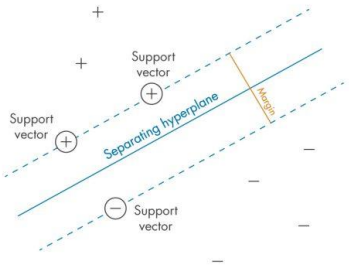
- Přípravná fáze před vlastním učením
- Výběr AI modelu / architektury sítě
- Nastavení možností pro učení
  - učící algoritmus, ztrátová funkce, hyperparametry
- Interaktivní nástroje nebo funkce
  - Deep Network Designer
  - Classification Learner
  - Regression Learner
- Hledání optimálních hyperparametrů
  - nastavení experimentů
  - Experiment Manager





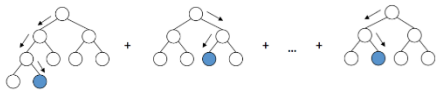
AI modely v prostředí MATLAB

Machine learning



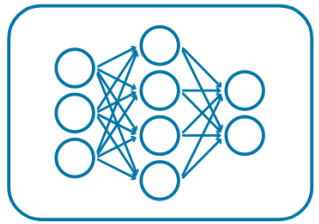
SVM

Clustering

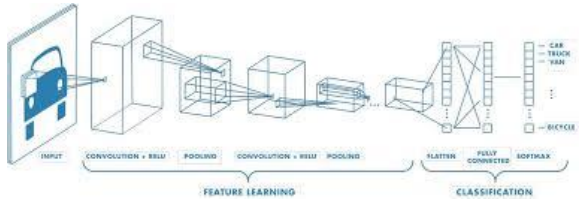


Decision trees

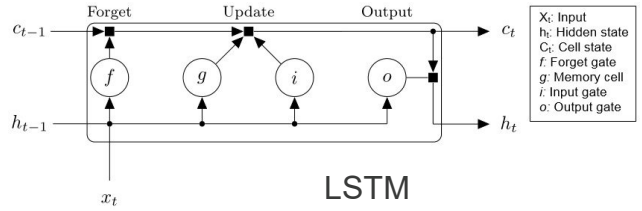
Deep learning



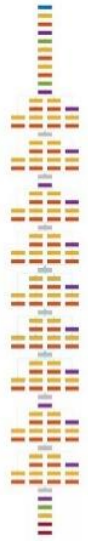
FC



CNN

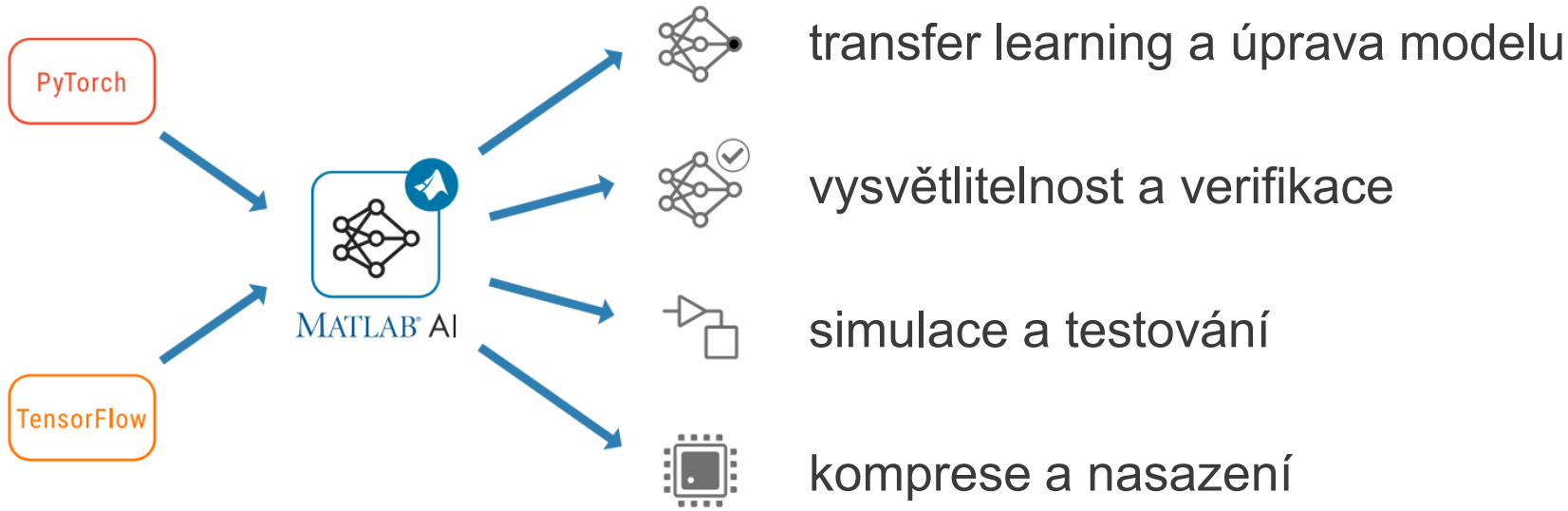


LSTM

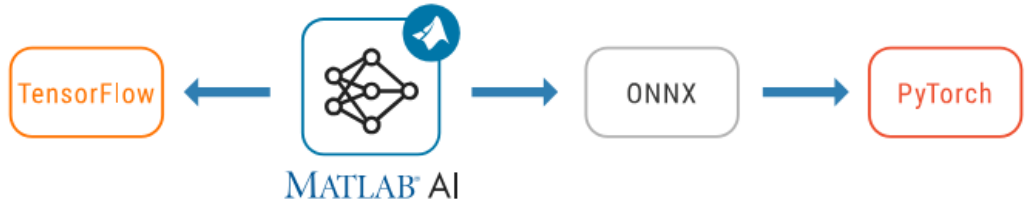


MATLAB a spolupráce s dalšími nástroji

- Import AI modelů z prostředí TensorFlow, PyTorch a v otevřeném formátu ONNX



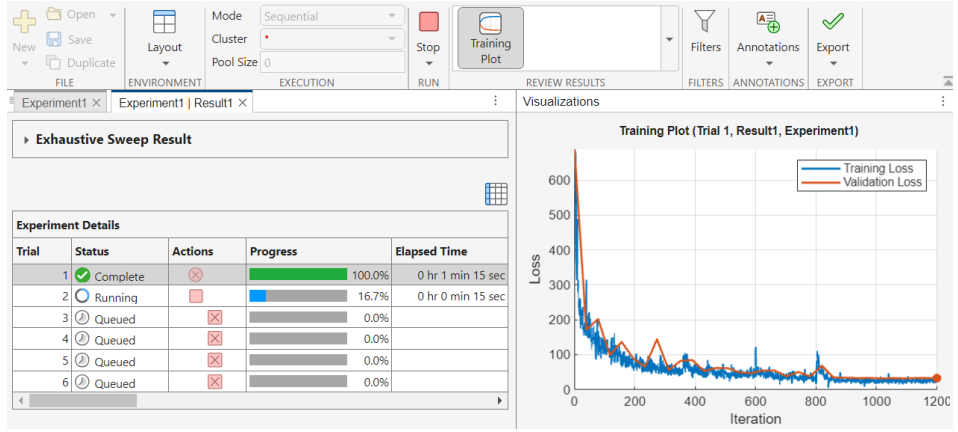
- Export AI modelů

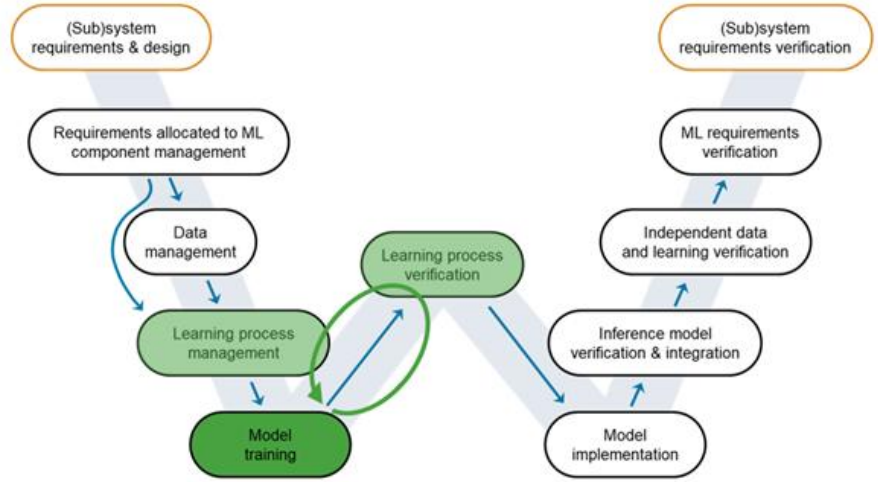


https://www.mathworks.com/campaigns/pocket-guides/matlab-python-ai/ebook.html

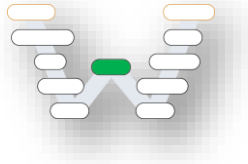
Učení AI modelu

- Učení zvoleného modelu / skupiny modelů
- Posouzení dosažené přesnosti modelu
- Hledání optimálních hyperparametrů
  - spuštění experimentů
  - výběr nejlepšího modelu
- Iterativní proces v rámci W-cyklu
  - změny nastavení v případě nedostatečných výsledků
  - výběr odlišných metod pro učení, …

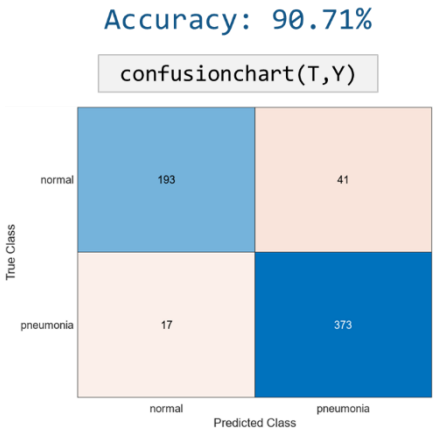


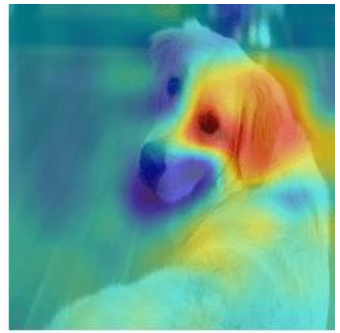


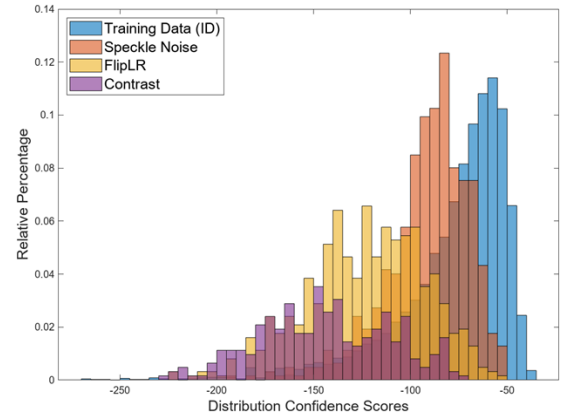
Verifikace procesu učení

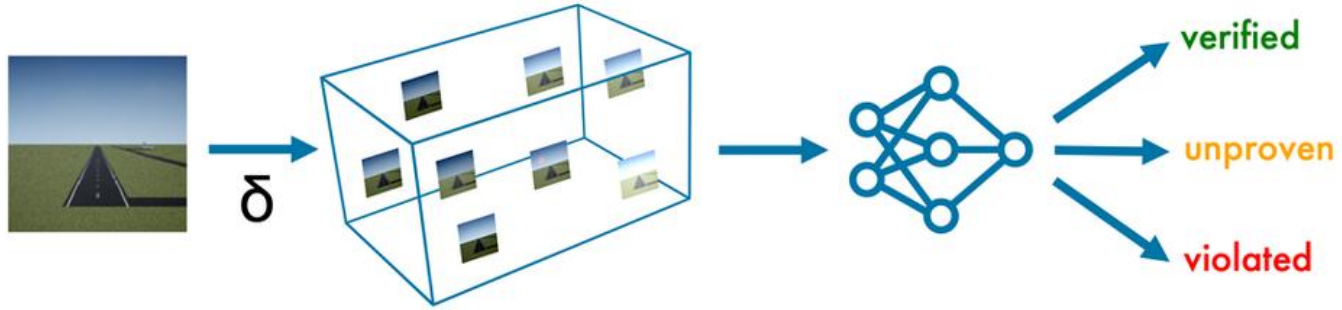


- Testování AI modelu
  - důležité je testovat s nezávislou sadou dat
  - metriky pro vyhodnocení úspěšnosti sítě
- Pochopení chování AI modelu (XAI)
  - interpretovatelnost modelu, vysvětlitelnost modelu
  - vizualizační techniky pro pochopení chování modelu
- Ověření robustnosti AI modelu
  - adversarial examples, formální verifikace
  - detekce „out-of-distribution“ dat

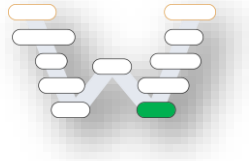




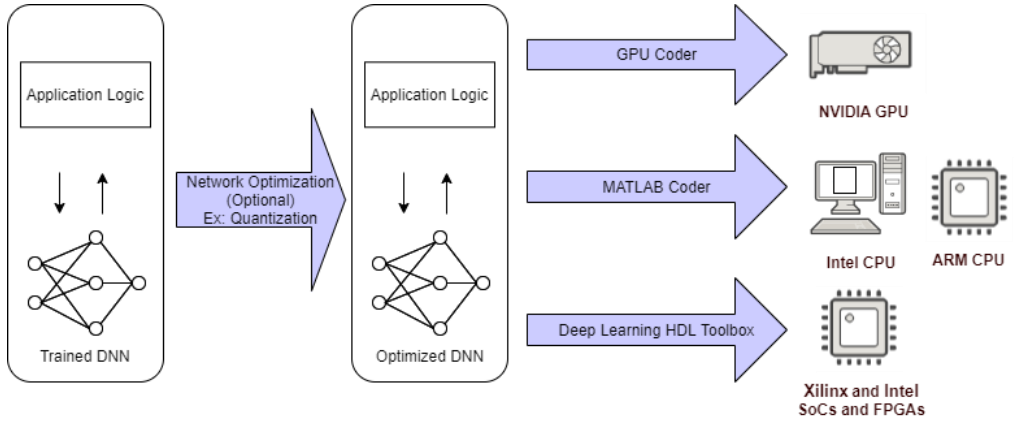


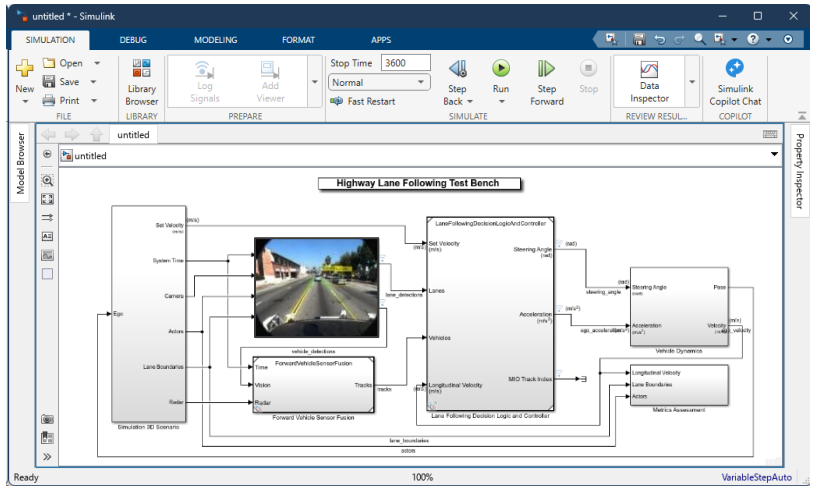


Implementace AI modelu



- Příprava AI modelu pro implementaci
  - komprese modelu (hluboké neuronové sítě)
- Automatické generování kódu
  - generování kódu z naučených AI modelů
  - optimalizovaný kód pro cílovou platformu
  - C/C++, CUDA, HDL
- Simulink jako platforma pro implementaci a integraci
  - implementace řešena současně s fází integrace a verifikace
  - propojení AI modelu s dalšími algoritmy (řídicí systém, …)
  - testování prostřednictvím simulace
  - společná implementace (automatické generování kódu)

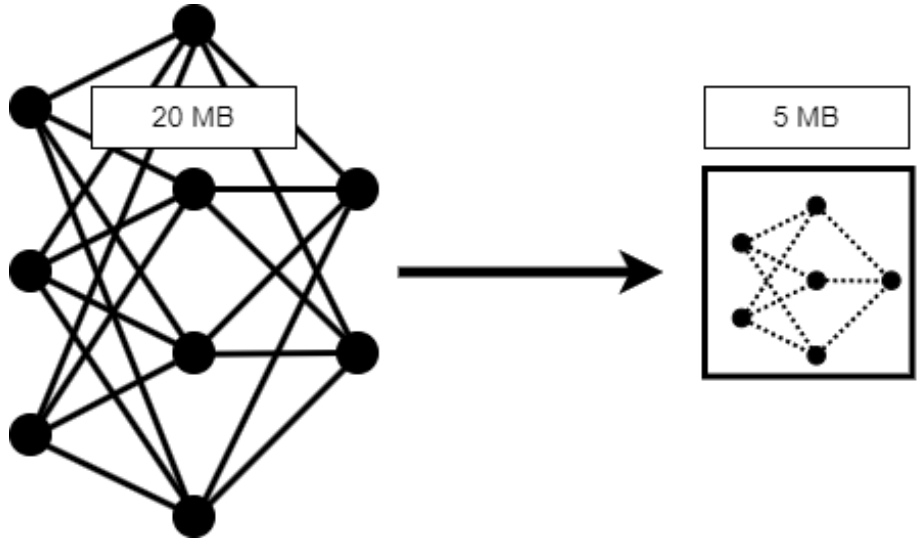




https://www.mathworks.com/help/deeplearning/code-generation.html

Komprese AI modelu

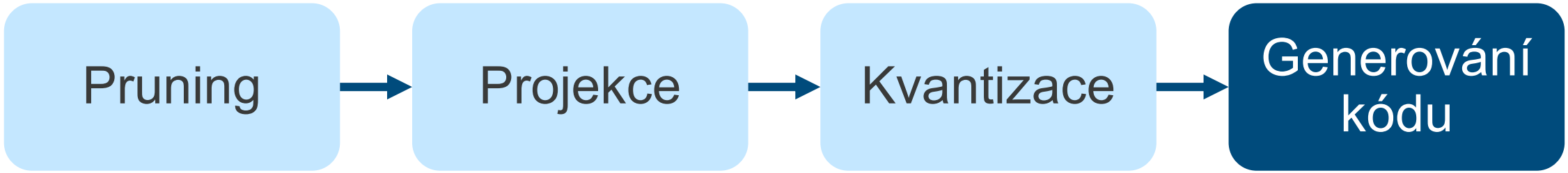
- Cíl: snížit paměťovou náročnost a výpočetní nároky hluboké neuronové sítě
- Dva hlavní faktory přispívající k velikosti neuronové sítě:
- Stavy
  - informace vypočítané během operací s vrstvami
  - uchovávají se pro použití v následných průchodech vrstvou
  - např. vnitřní stavy LSTM vrstev
- Učitelné parametry
  - obsahují vlastnosti, které se síť naučila
  - např. váhy konvolučních a plně propojených vrstev



https://www.mathworks.com/help/deeplearning/ug/reduce-memory-footprint-of-deep-neural-networks.html

Způsoby komprese AI modelu v prostředí MATLAB

- Strukturální komprese – pruning, projekce
  - snížení počtu stavů a učitelných parametrů
- Komprese datových typů – kvantizace
  - převod stavů a parametrů na datové typy s nižší přesností
- Kompresní techniky lze kombinovat ~ je dobré aplikovat ve tomto pořadí

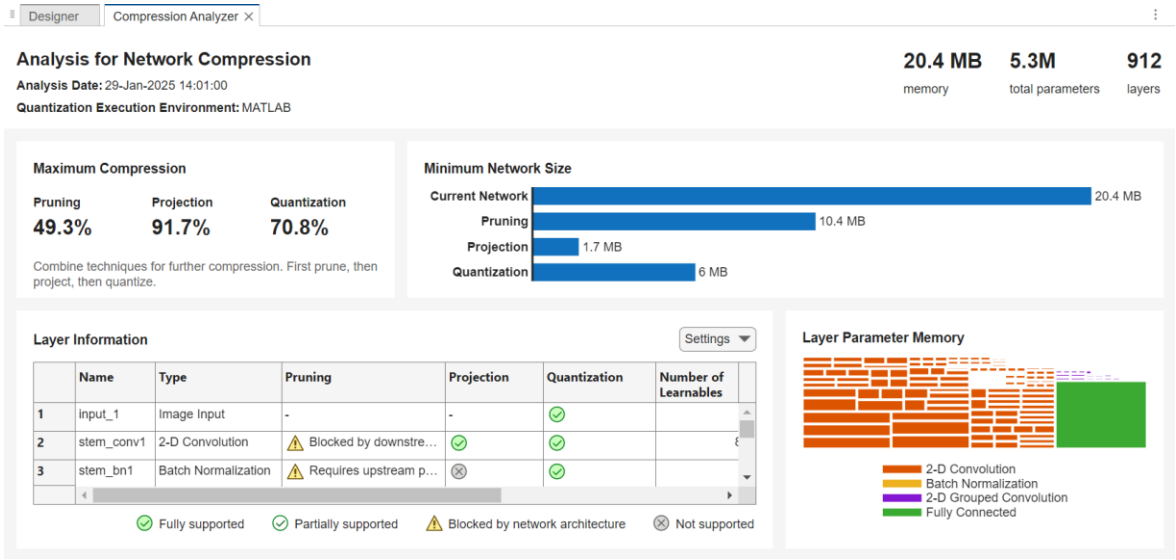


- K dosažení optimálních výsledků po každém kroku neuronovou síť dotrénovat

https://www.mathworks.com/help/deeplearning/ug/train-and-compress-ai-model-for-road-damage-detection.html

Analýza možností komprese AI modelu

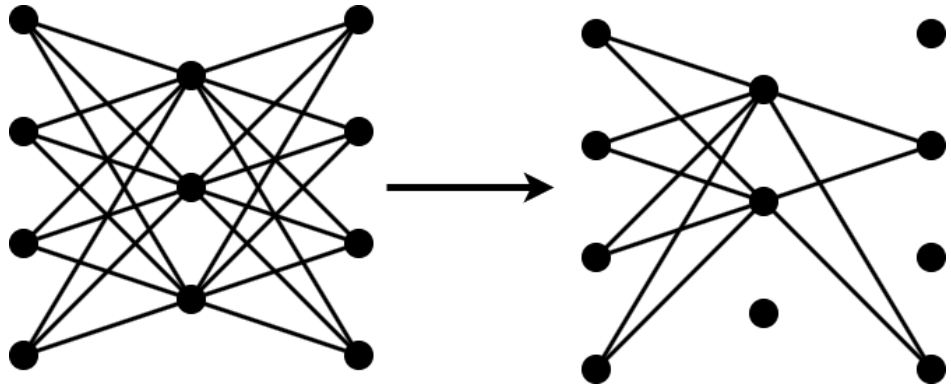
- Analýza zobrazuje informace:
  - maximální možná úspora paměti
  - podpora pruningu, projekce a kvantizace pro danou síť
  - omezení pruningu jednotlivých vrstev z důvodu architektury sítě



- Slouží k rozhodnutí, kterou techniku ke kompresi sítě použít

Pruning (prořezávání)

- Systematické odstranění nejméně důležitých parametrů sítě, aby se zmenšila velikost sítě a zároveň se zachovala kvalita jejích predikcí



- Měření významu skupiny parametrů
  - dle změny hodnoty ztrátové funkce po odstranění parametru
- Iterativní proces
  - 1. určení důležitosti filtrů, 2. odstranění nejméně důležitých filtrů, 3. doučení sítě
- Funkce (all-in-one): **`compressNetworkUsingTaylorPruning`**

https://www.mathworks.com/help/deeplearning/pruning.html

Taylor Pruning – na čem je založen

- Určení důležitosti parametrů pomocí Taylorova rozvoje (aproximace)
  - ztráta (loss) jako funkce jednotlivých parametrů sítě θ
  - odstranění parametru je ekvivalentní nastavení jeho hodnoty na 0

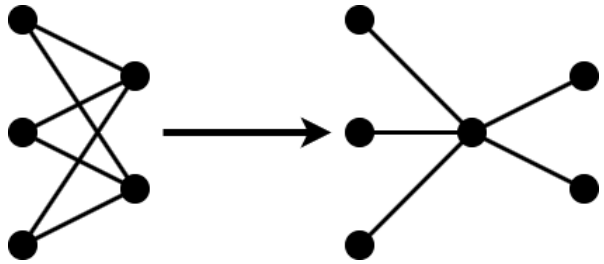
$$|\Delta loss(X, \theta)| = |loss(X, \theta = 0) - loss(X, \theta)|$$

- Taylorův rozvoj: $loss(X, \theta) = loss(X, \theta = 0) + \frac{\delta loss}{\delta \theta}\theta$
- Změna ztráty je funkce gradientu: $|\Delta loss(X, \theta)| = \left|\frac{\delta loss}{\delta \theta}\theta\right|$
- Gradient počítán při učení sítě $\Rightarrow$ využijeme je znovu

https://www.mathworks.com/help/deeplearning/ref/compressnetworkusingtaylorpruning.html

Projekce

- Způsob převodu velkých vrstev s mnoha učitelnými parametry na jednu nebo více menších vrstev s nižším celkovým počtem učitelných parametrů



- K redukci velikosti využívá PCA (principal-component analysis)
- Po provedení projekce následuje doučení sítě
  - k obnovení přesnosti predikce ztracené projekcí
- Funkce: **`compressNetworkUsingProjection`**

https://www.mathworks.com/help/deeplearning/projection.html

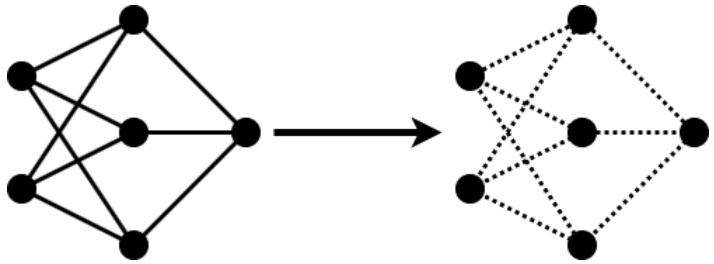
Projekce – na čem je založena

- V neuronových sítích jsou data reprezentována jako více-rozměrné vektory W
  - ne všechny dimenze jsou pro danou úlohu stejně důležité
- PCA umožňuje vyjádřit data v podobě dimenzí seřazených podle důležitosti
- Princip projekce: Wx nahradí za $WQQ^T x$
  - Q matice projekce do méně-rozměrného prostoru
  - vrstva (*projected layer*) ukládá Q a $W' = WQ$ namísto původní matice W
- Postup
  - 1. určí podprostor učitelných parametrů s nejvyšším rozptylem v aktivacích neuronů (PCA)
  - 2. promítne vstup vrstvy do prostoru nižší dimenze (N nejdůležitějších směrů)
  - 3. provede funkci dané vrstvy v prostoru nižší dimenze
  - 4. vrátí se do prostoru vyšší dimenze

https://www.mathworks.com/help/deeplearning/ref/compressnetworkusingprojection.html

Kvantizace

- Technika komprese datových typů, která snižujete numerickou přesnost parametrů v síti, ale množství parametrů a architektura sítě zůstávají stejné

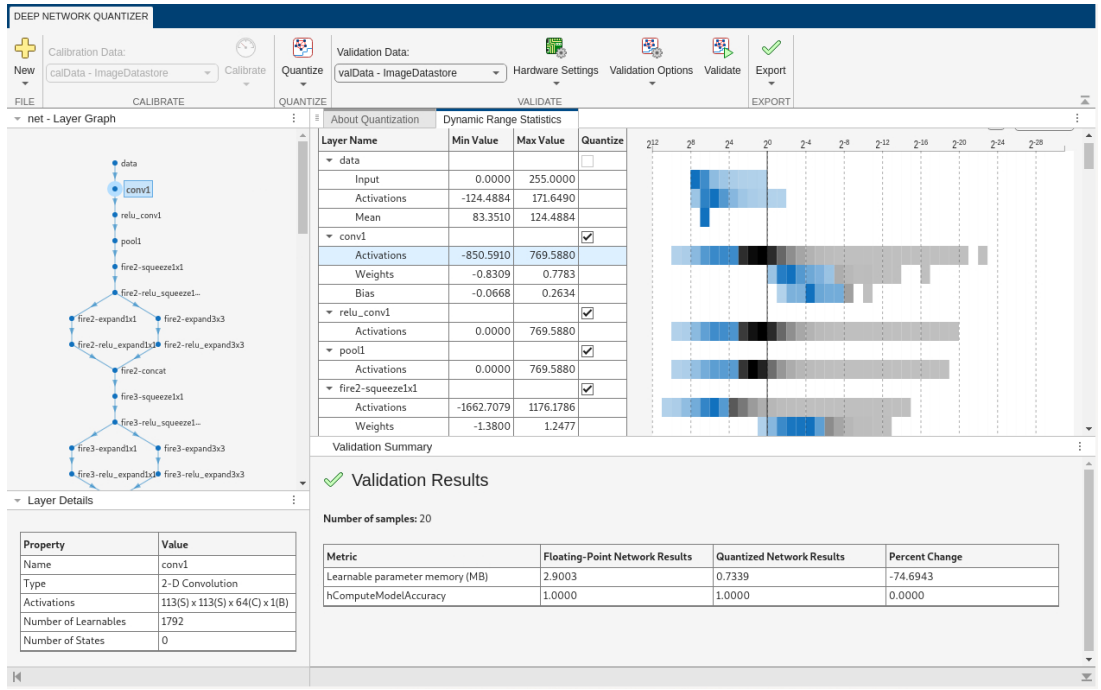


- Postup kvantizace
  - nalezení dynamických rozsahů parametrů sítě pomocí reprezentativního vzorku dat
  - převedení parametrů všech vrstev na celá čísla dle dynamického rozsahu z předchozího kroku
- Funkce (objekt): **`dlquantizer`**
  - postup: 1. vytvoření a příprava objektu, 2. kalibrace, 3. kvantizace, 4. validace
- Aplikace: **Deep Network Quantizer**

https://www.mathworks.com/help/deeplearning/quantization.html

Co poskytuje aplikace Deep Network Quantizer

- Vizualizace dynamických rozsahů konvolučních vrstev
- Výběr vrstev pro kvantizaci
- Posouzení chování kvantizované sítě
- Generování kódu
  - C++ pro ARM Cortex-A
  - GPU pro grafické akcelerátory
  - HDL pro FPGA

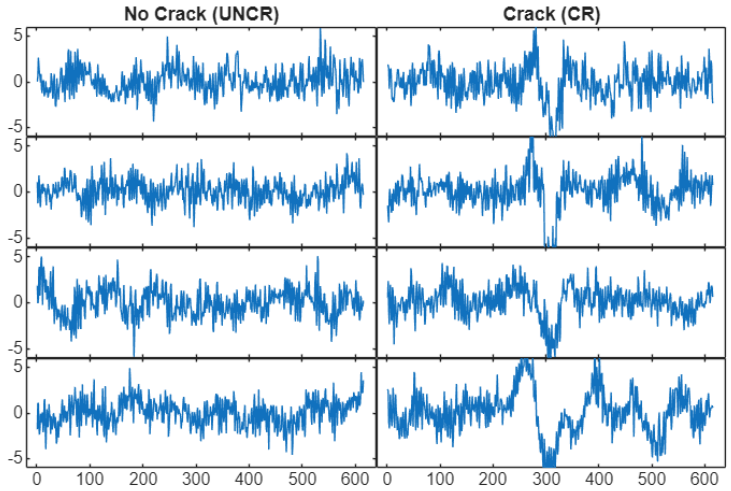


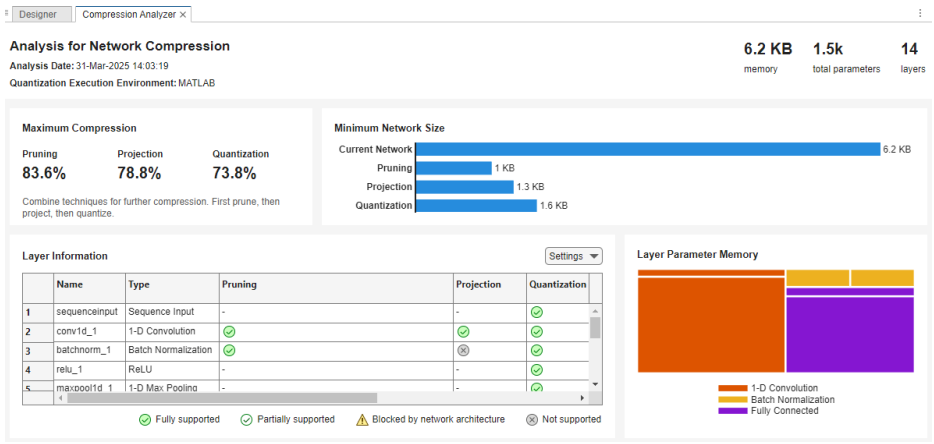
- Generování simulovatelné kvantizované síťě
  - lze prozkoumat v prostředí MATLABu bez generování kódu nebo nasazení na hardware

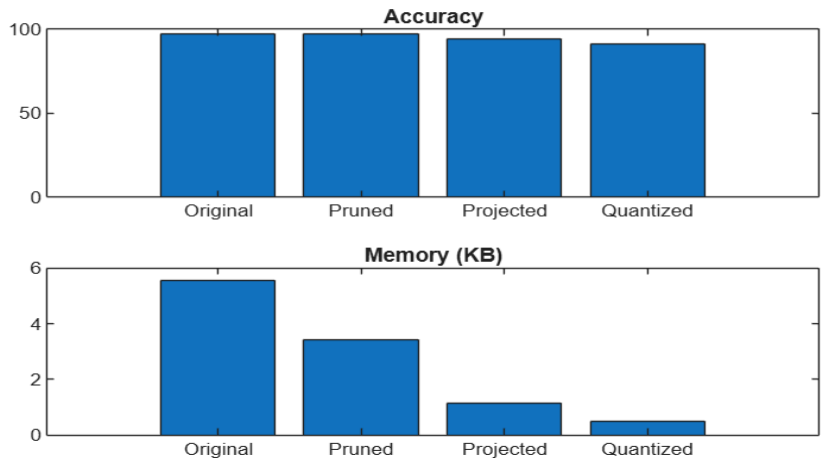
https://www.mathworks.com/help/deeplearning/ref/deepnetworkquantizer-app.html

Ukázka: Komprese AI modelu pro klasifikaci sekvencí

- AI model určený pro detekci vad na vozovce
  - typ modelu: neuronová síť, 1D konvoluční vrstvy
- Data z akcelerometru při jízdě po vozovce
  - přesnost detekce poškozených úseků: 97 %
- Komprese: pruning, projekce, kvantizace







https://www.mathworks.com/help/deeplearning/ug/train-and-compress-ai-model-for-road-damage-detection.html

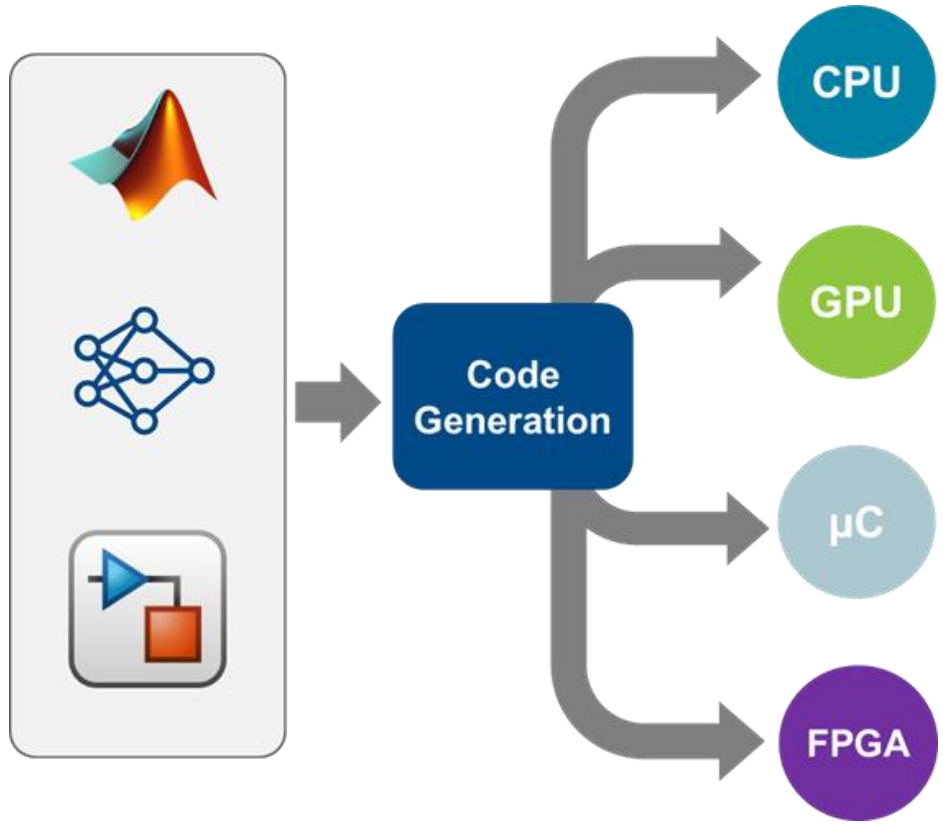
Další techniky

- Destilace znalostí (*knowledge distillation*)
  - využívá velkou a přesnou síť učitele k naučení menší sítě studenta
  - umožňuje navrhnout malou síť tak, aby vyhovovala paměťovým požadavkům
- Kvantizace pomocí knihovny NVIDIA TensorRT (GPU Coder)
  - kód pro 8bitovou celočíselnou predikci generovaný pomocí knihovny NVIDIA TensorRT
- Kvantizace pomocí kódu bfloat16 (MATLAB Coder)
  - bfloat16 je zkrácená verze formátu single
  - zabírá pouze 16 bitů (oproti 32), přibližně stejný rozsah čísel (stejný počet bitů exponentu)
- Řídkost
  - lze použít k prořezání sítě pomocí vlastní metriky důležitosti
  - nejméně důležité parametry nastavit na nulu a výsledné matice vah uložit jako řídké matice

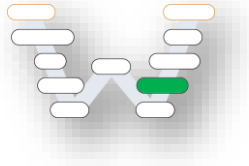
Reduce Memory Footprint of Deep Neural Networks - Other Techniques

Automatické generování kódu

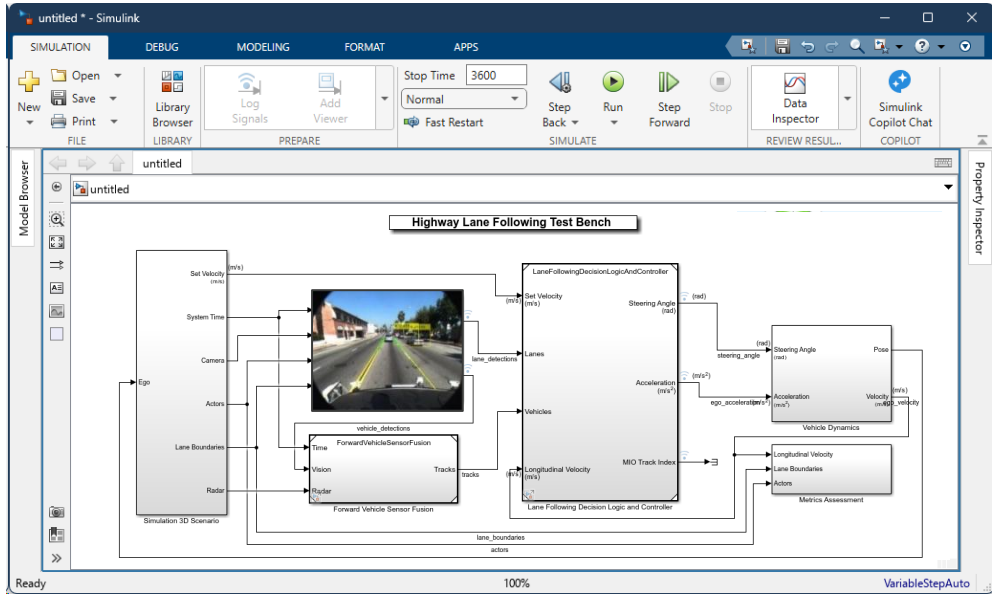
- Generování kódu z naučených a komprimovaných AI modelů
  - z prostředí MATLAB
  - z prostředí Simulink
- Jazyky a cílové platformy
  - C/C++ pro CPU a µP
  - CUDA pro GPU
  - HDL pro FPGA a SoC
- Nastavení generátoru kódu
  - vzhled a struktura kódu
  - optimalizace pro cílovou platformu
  - reporty a trasovatelnost



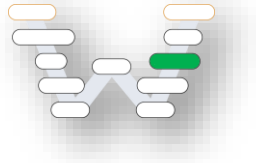
Verifikace a integrace modelu pro inferenci



- Dvě důležité provázané fáze nasazení AI modelu
  - přechod modelu z teoretické roviny do prakticky provozovaného nástroje
- Verifikace implementovaného modelu
  - ověření přesnosti modelu v implementované podobě
  - zásadní zejména pro bezpečnostně kritické aplikace
- Integrace modelu
  - zapojení modelu do širšího návrhu
  - propojení s dalšími algoritmy
  - testování v kontextu celkové funkčnosti
- Simulink jako platforma pro integraci
  - integrace bývá řešena současně s předchozí fází implementace
  - testování prostřednictvím simulací a nástrojů pro tvorbu a správu testů (Simulink test)

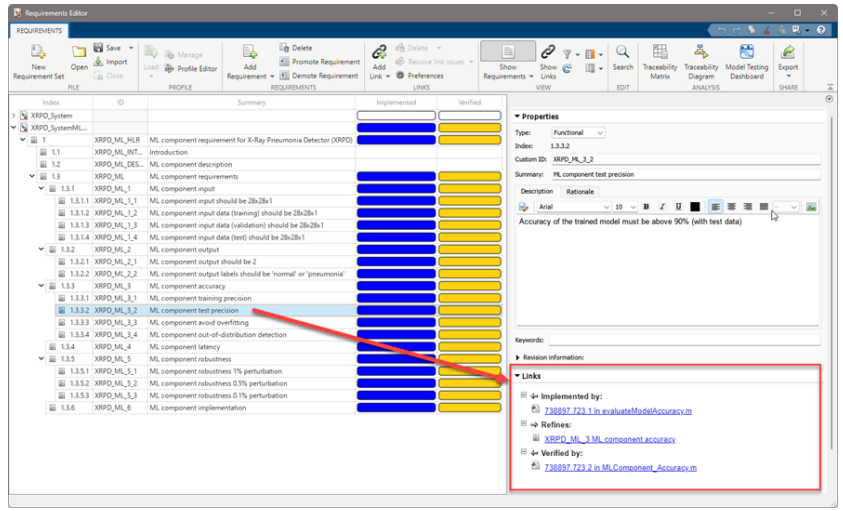


Nezávislé ověření dat a procesu učení

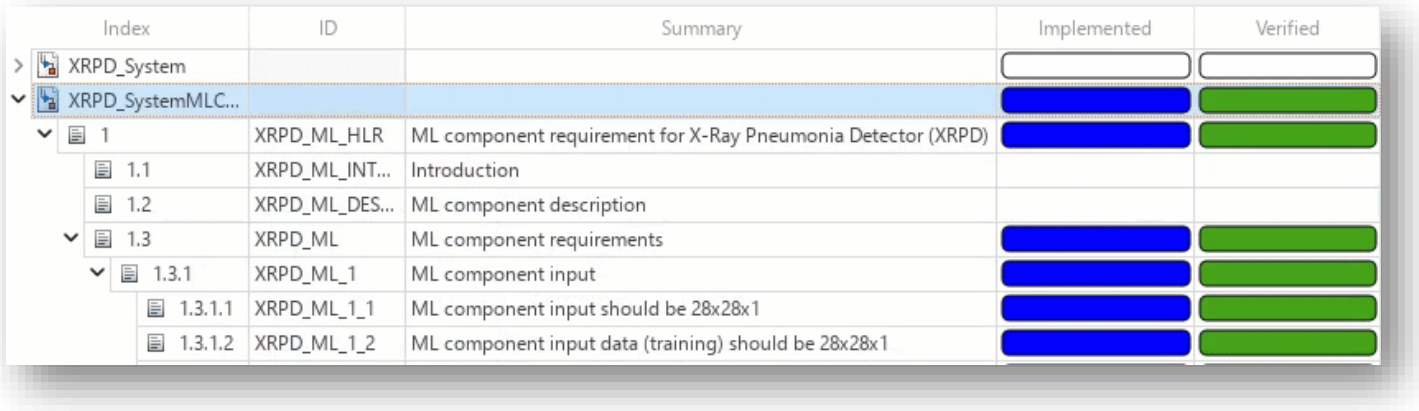


- Následuje po důkladném ověření inferenčního modelu na cílové platformě
- Ověření, zda byly datové sady v průběhu vývoje řádně spravovány
  - nezávislé přezkoumání které potvrdí, že trénovací, validační a testovací datové sady:
  - a) splňují striktní požadavky na správu dat
  - b) jsou úplné a reprezentativní pro vstupní datový prostor dané aplikace
- Ověření procesu učení
  - ověření, zda byl trénovaný model uspokojivě verifikován, včetně nezbytných analýz pokrytí

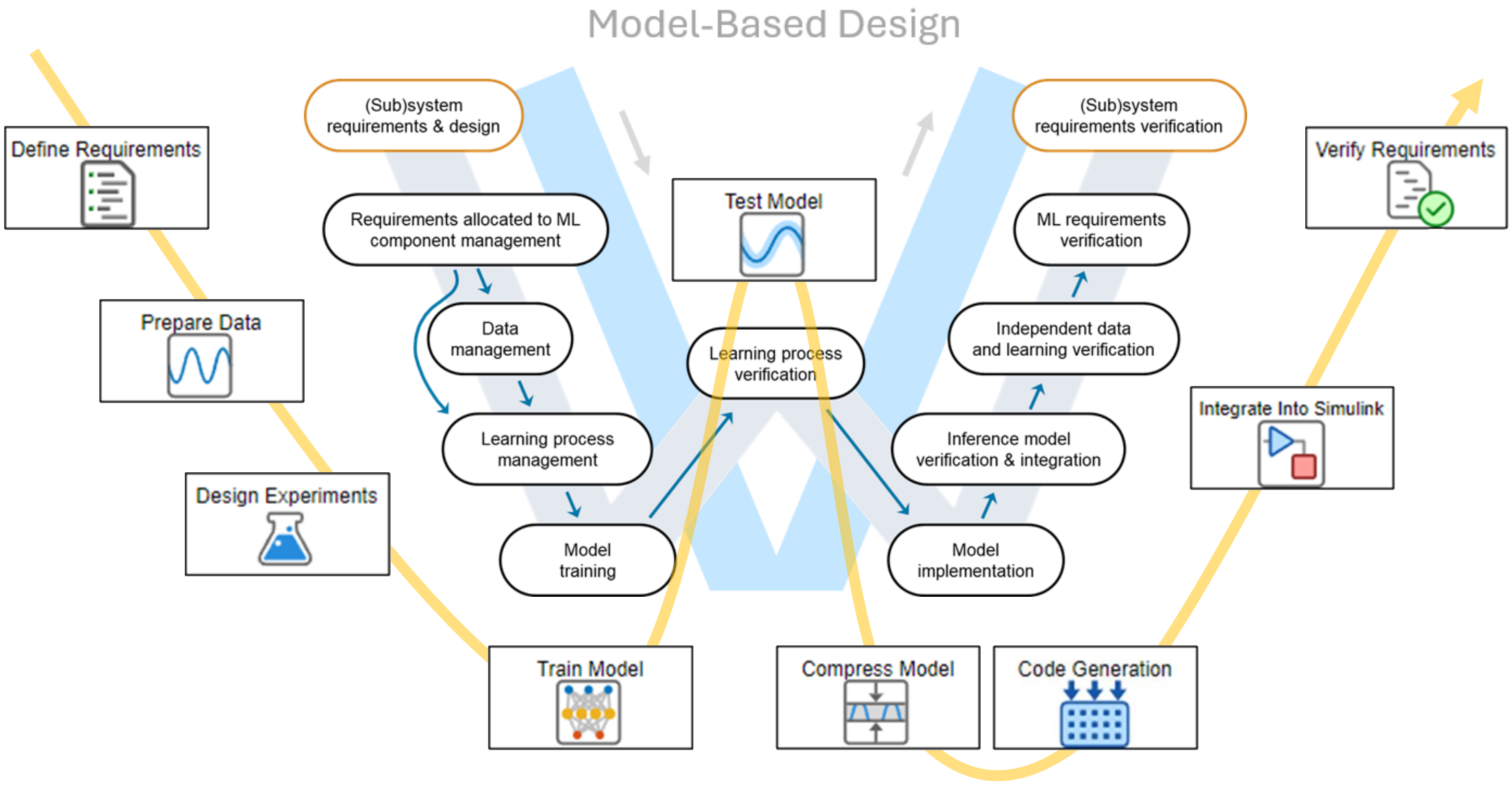
Verifikace požadavků



- Závěrečná část W-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



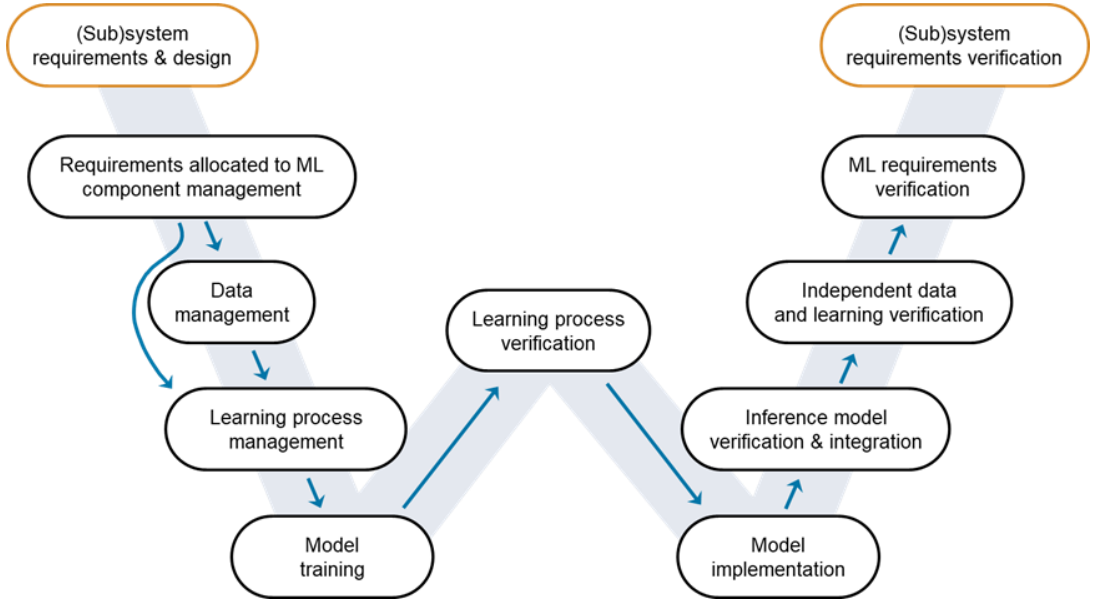
Shrnutí: Vývojový cyklus W ~ propojení AI a MBD



Další informace a příklady

- Série článků
  - část 1. úvod
  - část 2. požadavky a modelování
  - část 3. verifikace učení
  - část 4. implementace a verifikace požadavků

- Ukázky:
  - End-to-End AI Workflows



Credit: EASA, Daedalean.

Vybrané uživatelské reference

- AI-Powered Virtual Sensor Delivers Real-Time Cabin Air Mass Flow Estimation
  - Mercedes-Benz Research & Development
- TinyML to Enhance Performance of Field-Oriented Control
  - STMicroelectronics
- Implementing a Workflow for Deploying and Integrating Deep Learning Networks on PLCs for Industrial Automation
  - Beckhoff Automation GmbH & Co. KG

Děkuji za pozornost

Kontakty

- 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