Témata magisterské státní závěrečné zkoušky

Cílem tohoto textu je rozvést stručné znění témat magisterské státní závěrečné zkoušky tak, aby bylo zřejmé (1) jaké konkrétní znalosti patří do obecně uvedených témat, (2) v jaké hloubce budou tyto znalosti vyžadovány, a (3) jakými předměty jsou pokryty.

Tato verze je připravená pro magisterské státní závěrečné zkoušky od června 2022.

Softwarové systémy: Spolehlivé systémy

> Toto zaměření ověřuje znalosti a dovednosti týkající se konstrukce spolehlivých softwarových systémů, zkoušené v rozsahu následujících profilujících předmětů.

Detailní požadavky

NSWI132 Analýza programů a verifikace kódu

- Vysvětlit různé metod pro verifikaci a analýzu chování programů a softwarových systémů (model checking, statická analýza, deduktivní metody, jejich kombinace)
- Popsat výhody a nevýhody metod verifikace a analýzy chování programů zejména vzhledem ke testování
- Důležité aspekty: přesnost, škálovatelnost, výkonnost (rychlost)
- Metody pro detekci chyb především ve vícevláknových programech
  - Popsat metody založené na průchodu stavovým prostorem: bounding the number of preemptions, vybrané heuristiky
  - Popsat metody založené na dynamické analýze programů (lock-set analýza, happens-before relation)
  - Definovat stav programu a jeho reprezentace, stavový prostor (jeho součásti)
  - Definovat pojem korektnosti vícevláknových programů (data-race freedom, serializability, linearizability, ABA problém)
  - Popsat relaxed/weak memory models z perspektivy korektního chování programů a hledání chyb
- Symbolic execution (vysvětlit základní princip, popsat důležité praktické aplikace)
- Abstrakce chování programů
  - Uvést příklady nějakých abstrakcí, vysvětlit koncept predikátové abstrakce
  - Vysvětlit pojmy over-approximation a under-approximation v kontextu abstrakce chování programů
  - Vysvětlit důsledky použití over/under-aproximace s ohledem na ověření vlastnosti a nalezení chyb
- Deduktivní metody verifikace chování programů
  - Popsat způsob reprezentace chování programu a jeho vlastností (invariantů, kontraktů, preconditions a postconditions) ve formě logické formule
  - Vysvětlit pojem "verification condition (VC)" a základní proces generování VC
- Statická analýza programů
  - Vysvětlit základní koncepty (control-flow graph, jak se reprezentují informace o analyzovaném programu, reprezentace sémantiky příkazů/instrukcí a jejich vlivu na výsledky analýzy, worklist algorithm)
  - Základní klasifikace statických analýz (direction, scope, level of approximation ("may" versus "must"), flow-sensitivity, context-sensitivity)

NTIN043 Formální základy softwarového inženýrství

- Srovnat jednotlivé metody formální specifikace požadavků, chování a architektury softwarových systémů vzhledem ke vyjadřovací síle, metodologii, a vhodné aplikační doméně
  - Metody: algebraické specifikační jazyky, modelově-orientované specifikační jazyky (Z, VDM, Alloy), UML a OCL, Petriho sítě
- Obecná charakteristika formálních metod (účel, výhody, nevýhody) a způsob jejich použití
- Algebraické specifikační techniky

  - Popsat základní principy a dílčí prvky specifikace (sort, carrier set (nosič), operations and functions, equations)
  - Definovat pojem sémantiky algebraické specifikace (na jaké úrovni tyto specifikace popisují chování software, jaké vlastnosti umožňují vyjádřit)
  - Vysvětlit související pojmy: executable specifications, rewriting system (včetně důležitých a potřebných vlastností, jako "confluence", "termination" a "canonical form")
- Modelově-orientované specifikace
  - Vysvětlit základní principy: jak se vytváření specifikace, důležité prvky a vlastnosti
  - Popsat metodologie vytváření specifikace
  - Popsat iterativní zjemňování (refinement) v kontextu modelově-orientovaných jazyků
- UML a OCL
  - Popsat hlavní typy UML diagramů (class, sequence, activity, use cases)
  - Popsat základní výrazové prostředky OCL (výrazy, preconditions, postconditions, invariants)
- Petriho sítě
  - Popsat základní koncepty (place, transition, arc, token, configuration/marking, firing of enabled transitions)
  - Vytvořit příklad modelu pro jednoduchý systém (například mutual exclusion dvou vláken)
- Aplikace formálních metod v praxi vývoje software
  - Vysvětlit pojmy Model-driven engineering, model-based testing, generating code from specification (iterative refinement)

NSWI164 Modelem řízený vývoj

- Vysvětlit (s příklady) co je model-driven development.
- Vysvětlit, co je model a dát příklad. Vysvětlit, co je concrete syntax a dát příklad (textová, grafická)
- Vysvětlit, co je meta-model a k čemu se používá. Být schopen nakreslit meta-model (pomocí UML class diagramů) ke konkrétnímu zadání (např. modelování tříd a interfaců zvoleného OO jazyka)
- Vysvětlit a dát příklad k meta-modelovací hierarchii (model, meta-model, meta-meta-model, …)
- Vysvětlit, co je domain-specific language a dát příklad. Ukázat, jak se dá DSL specifikovat (např. pomocí Xtextu)
- Vysvětlit, co je transformace modelů a dát příklad ve zvoleném jazyce
- Vysvětlit, co je generování kódu a jak se dá realizovat pomocí nástrojů k tomu určených

NSWI101 Modely a verifikace chování systémů

- Formalismy pro reprezentaci modelů počítačových systémů a jejich vlastnosti
  - Definovat pojmy Kripkeho struktura, LTS, State-transition system, uvést příklady těchto struktur
  - Vyjmenovat a popsat některé jazyky pro specifikaci těchto struktur, popsat vybraný přístup spolu s praktickými aspekty (v jakých případech je vhodné daný formalismus použít) – Promela, Timed Automata, Parallel-assignment language (SMV), TLA+/PlusCAL
- Temporální logiky pro specifikaci vlastností
  - Definovat syntax a sémantiky logik CTL, LTL, TimedCTL, ICTL, …
  - Porovnat logické síly LTL a CTL, příklady dokazující neporovnatelnost těchto logik
- Algoritmy pro model checking
  - Popsat algoritmus pro model checking LTL, CTL, TimedCTL (Timed Automata), popsat vybraný algoritmus
  - Ukázat asymptomatickou složitost těchto algoritmů
- Symbolický model checking
  - Definovat OBDD a popsat tento modelovací přístup, popsat složitost logických operací nad OBDD
  - Popsat algoritmus pro symbolický model checking CTL
- Specifické přístupy k verifikaci a model checkingu
  - Vysvětlit pojmy Infinite a Bounded model checking, abstrakce, probabilistic model checking – popsat jednotlivé metody, uvést motivaci jejich existence
  - Uvést příklady aplikací spolu s příklady vlastností, které pomocí těchto logik lze verifikovat

NSWE001 Vestavěné systémy a systémy reálného času

- Charakteristika embedded a real-time systémů

  - Vysvětlit, co je specifické pro embedded systém
  - Vysvětlit, co je specifické pro real-time systém
  - Charakterizovat různé typy real-time systémů pomocí utility funkce
- Úlohy a plánování
  - Vysvětlit, co je real-time úloha a rozdělit dle typů (time/event triggered, periodická/aperiodická)
  - Vyjmenovat a vysvětlit parametry úloh (arrival time, computation time, relative/absolute deadline, lateness, jitter, …)
  - Definovat problém plánování (včetně omezujících podmínek – blokování, precedence)
- Aperiodické plánování
  - Vysvětlit a demonstrovat algoritmy pro plánování při synchronní/asynchronní aktivaci, preemptivním/nepreemptivním plánování, s/bez precedence constraints
  - Vždy vyjmenovat a vysvětlit předpoklady algoritmů
- Periodické plánování
  - Vysvětlit a demonstrovat algoritmy pro offline/online plánování, u online plánování dále se statickými/dynamickými prioritami
  - Vysvětlit jak se dělá u jednotlivých algoritmů záruka plánovatelnosti (schedulability guarantee) – prohledávání u offline alg., Liu-Leyland bound a response time analysis u RM a DM, Liu-Leyland bound a processor demand analysis u EDF
  - Vysvětlit, co znamená optimalita u RM/DM a co znamená u EDF
  - Vysvětlit, jak se počítá response time analysis s jitter, blokováním a overheadem systémů
- Plánování s blokováním
  - Vysvětlit, co je blokování a jak vzniká
  - Vysvětlit, jak se počítá response time analysis, když je v systému blokování
  - Vysvětlit a demonstrovat (nakreslit rozvrh) priority inversion problem
  - Vysvětlit a demonstrovat algoritmy PIP a PCP. U obou algoritmů říct, co za problémy řeší, co neřeší a jaké dávají záruky.
- Design
  - Vysvětlit módy systému
  - Popsat/demonstrovat, jak se získávají real-time požadavky na úlohy ze specifikace
- Controllers
  - Vysvětlit, co je PID, jeho složky a praktický význam jednotlivých složek pro řízení
- Komunikace
  - Rozdíl time-triggered a event-triggered komunikace
  - Popsat fungování CANu a zajištění real-time garancí
  - Vysvětlit a demonstrovat způsob, jak se dělá schedulability analysis v distribuovaném systému s CANem
- Servery
  - Vysvětlit, co a k čemu jsou servery v real-time systémech
  - Rozdíl mezi static a dynamic priority servery
  - Vysvětlit a demonstrovat algoritmy pro static a dynamic priority servery – polling server, deferrable server, priority exchange server, sporadic server, dynamic priority exchange server, dynamic sporadic server, total bandwidth server, constant bandwidth server, IPE

Pokryto předměty

- NSWI132 Analýza programů a verifikace kódu
- NTIN043 Formální základy softwarového inženýrství
- NSWI164 Modelem řízený vývoj
- NSWI101 Modely a verifikace chování systémů
- NSWE001 Vestavěné systémy a systémy reálného času

Softwarové systémy: Systémové programování

Toto zaměření ověřuje znalosti a dovednosti týkající se systémového programování a vnitřní funkce softwarových systémů, zkoušené v rozsahu profilujících předmětů.

Detailní požadavky

NPRG014 Koncepty moderních programovacích jazyků

- Vysvětlit a ilustrovat na příkladu rozdíl mezi class-based jazyky se statickým typováním, dynamickým typováním a prototypovými jazyky (tj. jazyky bez tříd)
- Class-based jazyky – vysvětlit a ilustrovat na příkladech ve zvoleném jazyce následující
  - Koncepty tříd, dědičnosti, interfaců, viditelnosti
  - Statické metody/fieldy vs. singletony (object vs. class ve Scale)
  - Uzávěry v typové hierarchii (Any, Null, Nothing). Ukázat, kdy se hodí.
  - Typové parametry (v deklaraci třídy a metod) a typové proměnné (tj. uvnitř tříd). Ukázat, jak se používají ve spojení s dolními a horními mezemi.
  - Kovarianci, kontravarianci a invarianci. Ukázat u každé (pomocí příkladu), kdy se hodí.
  - Koncepty traitů, mixinů a linearizace
  - Extension metody a implicitní konverze. Řici k čemu jsou potřeba.
  - Path-dependent typy
  - Implicitní parametry (typeclasses)
  - Meta-třídy
  - Koncepty funkcionálního programování v imperativních OO jazycích (currying, partial application, functors, monoids, tail recursion)
  - Vlastnosti jazyků umožňující interní DSL
  - Case-classes, pattern matching, extractors (metody apply a unapply)
  - Comprehensions
  - Statické meta-programování (makra/compiler plugins)
- Vysvětlit a ukázat na příkladu high-level abstrakce pro concurrency - dataflow, promises, actors, fork/join, geometric decompositions of collections, agents
- Prototypové jazyky – vysvětlit a ilustrovat na příkladech ve zvoleném jazyce následující
  - Koncept objektu bez třídy, sloty, parent sloty
  - Realizace konceptů jazyků se třídami v prototypovém jazyku (tj. nakreslit hierarchii objektů) – inheritance, statické membery/fieldy, instance fieldy

NSWI080 Middleware

- Volání vzdálených funkcí (RPC)
  - Na příkladech kódu demonstrovat mechanismus volání vzdálených funkcí vlastní volby (například RMI, gRPC, Thrift …)
  - Vysvětlit motivaci pro existenci jazyků pro definici rozhraní
  - Vysvětlit a na příkladech kódu demonstrovat funkci mechanismu thread pool
  - Vysvětlit funkci předávání referencí v objektových variantách volání vzdálených funkcí
- Předávání zpráv (messaging)
  - Vysvětlit motivaci pro existenci jazyků pro definici datových formátů
  - Diskutovat varianty technických parametrů reprezentace dat (binární vs textová, bez vs se schématem, bez vs s typovou informací)
  - Na příkladech kódu demonstrovat mechanismus publish-subscribe vlastní volby (například ActiveMQ, RabbitMQ, 0MQ …)
  - Na příkladech kódu demonstrovat mechanismus komunikace proudem zpráv (Kafka)
  - Na příkladech kódu demonstrovat mechanismus kolektivní komunikace (MPI)
- REST
  - Vysvětlit motivaci pro specifická pravidla architektonického stylu REST (reprezentace prostředků, manipulace s prostředky, absence stavu)
  - Na příkladech kódu demonstrovat definici rozhraní (OpenAPI)
- SOA a microservices
  - Vysvětlit motivaci

NSWI161 Pokročilé operační systémy

- Architektura operačního systému
  - Vysvětlit roli operačního systému a vyjmenovat a vysvětlit roli jeho základních komponent
  - Vysvětlit roli privilegovaného režimu procesoru a uvést příklady privilegovaných operací
  - Microkernels
  - Vysvětlit motivaci microkernel a unikernel architektur
  - Virtualizace
  - Vysvětlit motivaci virtualizace operačního systému
  - Vysvětlit rozdíl mezi virtuálním strojem a kontejnerem
  - Na běžícím kontejneru demonstrovat použití namespace objektů k dosažení izolace (lsns)
- Správa paměti
  - Heap
  - Vysvětlit a na příkladech kódu demonstrovat problémy alokace paměti na počítačích s více procesory
  - Ilustrovat řešení těchto problémů na konkrétním moderním alokátoru vlastní volby (například jemalloc, tcmalloc, mimalloc …)
  - Vysvětlit motivaci slab alokátoru a na daném příkladu kódu vysvětlit jeho použití
  - Stránkování
  - Na daném diagramu vysvětlit postup překladu adres ve virtuálních strojích na procesorech s hardwarovou podporou virtualizace (Intel EPT, AMD NPT)
- Systémy souborů
  - Vysvětlit mechanismy mirroring a striping a roli parity v konfiguracích RAID a jejich vliv na výkon a spolehlivost
  - Použít standardní nástroje na nastavení zadané RAID konfigurace (mdraid, btrfs)
  - Použít standardní nástroje na vytvoření nového svazku a obrazu svazku (btrfs)
  - Vysvětlit použití konceptu copy-on-write při vytváření obrazů svazků
- Správa sítě
- Správa služeb
  - Vysvětlit roli správy služeb v operačním systému
  - Použít standardní nástroje pro identifikaci a ovládání běžících služeb (systemctl)
  - Vytvořit k danému programu definici služby a jejích závislostí (systemd service unit file)
- Správa a vývoj operačního systému
  - Použít standardní nástroje ke sledování událostí uvnitř operačního systému (perf)
  - Vytvořit krátké programy (5-10 řádek) k monitorování provozu operačního systému (bpftrace)

NPRG058 Pokročilé programování v paralelním prostředí

- Multicore-CPU a NUMA systémy
  - Stručně popsat architekturu CPU a paměti (tj. klasický shared memory system) a kriticky je zhodnotit z pohledu efektivity paralelních algoritmů (CPU komunikace, sdílené cache, MESI protkol, NUMA faktor)
  - Na příkladech ukázat důsledky základních synchronizačních postupů na efektivitu algoritmů (context switch overhead, spin-lock starvation, false sharing a cache-line ping-pong…)
  - Navrhnout koncept lock-free implementace jednoduché datové struktury (linked list, hash table, growing vector…).
  - Navrhneout postup paralelizace jednoduchého algoritmu (násobení matic, k-means, …) s ohledem na cache a NUMA faktor.
- GPU
  - Na vhodném příkladu (implementace algoritmu) popsat princip použití datově-paralelního paradigmatu na GPU s poukazem na výhody a nevýhody lockstep exekuce
  - Na příkladu datově-agregačního algoritmu (např. výpočet histogramu) ukázat vhodné optimalizační techniky pro odstranění synchronizačních kolizí (privatizace, použití shared memory, paralelní redukce)
  - Ukázat vhodný příklad použití warp-kooperativních nebo warp-synchronizačních instrukcí pro optimalizaci kódu.

    - Navrhnout algoritmus (a popsat nejpodstatnější technické prostředky) pro optimalizaci přesunů dat mezi hostem a GPU; tento algoritmus porovnat s alternativami jako memory mapping nebo unified memory
    - Na vhodném algoritmu (např. Mandelbrot) ukázat možné výhody dynamického paralelismu; vysvětlit nezbytné technické prostředky, které musí GPU podporovat (context switch bloku vláken)
- MPI

NSWI035 Principy distribuovaných systémů

- Synchronizační algoritmy
    - Vysvětlit pojmy fyzické a logické hodiny, popsat algoritmus volby koordinátora a pojmy kauzální závislost, popsat doručovací protokoly
- Distribuovaný konsensus
    - Definovat pojem globálního stavu, popsat jeho detekci a jeho aplikace, popsat algoritmy pro dosažení distribuovaného konsensu, Paxos, RAFT
- Distribuovaná sdílená paměť
    - Popsat konzistenční modely, popsat a vysvětlit distribuované stránkování
- Správa prostředků a procesů
    - Definovat a vysvětlit distribuované algoritmy detekce zablokování

Pokryto předměty

- NPRG014 Koncepty moderních programovacích jazyků
- NSWI080 Middleware
- NSWI161 Pokročilé operační systémy
- NPRG058 Pokročilé programování v paralelním prostředí
- NSWI035 Principy distribuovaných systémů

Softwarové systémy: Výkonné systémy

Toto zaměření ověřuje znalosti a dovednosti týkající se konstrukce softwarových systémů s vysokým výpočetním výkonem, zkoušené v rozsahu následujících profilujících předmětů.

Detailní požadavky:

NSWI109 Konstrukce překladačů

- Popsat základní pojmy překladačů a reprezentace překládaného programu
  - Control flow, data flow, závislosti, aliasy, rozsahy platnosti
  - Sekvenční a nesekvenční mezikódy, SSA forma
- Popsat architekturu překladače, pořadí důležitých kroků překladu
- Alokace registrů
  - Popsat a vysvětlit algoritmus barvení grafu, řešení spill-kódu
- Scheduling
  - Popsat model procesoru, list scheduling, branch-and-bound přístup
  - Popsat algoritmy Unroll-and-compact, modulo scheduling, trace scheduling
- Paralelizace a vektorizace překladačem
  - Popsat polyhedrální kompilaci
- Analýza ukazatelů a aliasů
  - Popsat pojmy Andersenova a Steensgaardova metoda

NPRG058 Pokročilé programování v paralelním prostředí

- Multicore-CPU a NUMA systémy
  - Stručně popsat architekturu CPU a paměti (tj.klasický shared memory system) a kriticky je zhodnotit z pohledu efektivity paralelních algoritmů (CPU komunikace, sdílené cache, MESI protkol, NUMA faktor)
  - Na příkladech ukázat důsledky základních synchronizačních postupů na efektivitu algoritmů (context switch overhead, spin-lock starvation, false sharing a cache-line ping-pong…)
  - Navrhnout koncept lock-free implementace jednoduché datové struktury (linked list, hash table, growing vector…).
  - Navrhnout postup paralelizace jednoduchého algoritmu (násobení matic, k-means, …) s ohledem na cache a NUMA faktor.
- GPU
  - Na vhodném příkladu (implementace algoritmu) popsat princip použití datově-paralelního paradigmatu na GPU s poukazem na výhody a nevýhody lockstep exekuce
  - Na příkladu datově-agregačního algoritmu (např. výpočet histogramu) ukázat vhodné optimalizační techniky pro odstranění synchrnonizačních kolizí (privatizace, použití shared memory, paralelní redukce)
  - Ukázat vhodný příklad použití warp-kooperativních nebo warp-synchronizačních instrukcí pro optimalizaci kódu.
  - Navrhnout algoritmus (a popíše nejpodstatnější technické prostředky) pro optimalizaci přesunů dat mezi hostem a GPU; tento algoritmus porovnat s alternativami jako memory mapping nebo unified memory
  - Na vhodném algoritmu (např. Mandelbrot) ukázat možné výhody dynamického paralelismu; vysvětlit nezbytné technické prostředky, které musí GPU podporovat (context switch bloku vláken)
- MPI

NSWI035 Principy distribuovaných systémů

- Meziprocesová komunikace
  - Definovat a popsat pojmy spolehlivost, RPC, skupinová komunikace
- Synchronizační algoritmy

  - Vysvětlit pojmy fyzické a logické hodiny, popsat algoritmus volby koordinátora a pojmy kauzální závislost, popsat doručovací protokoly
- Distribovaný konsensus
  - Definovat pojem globálního stavu, popsat jeho detekci a jeho aplikace, popsat algoritmy pro dosažení distribuovaného konsensu, Paxos, RAFT
- Distribuovaná sdílená paměť
  - Popsat konzistenční modely, popsat a vysvětlit distribuované stránkování
- Správa prostředků a procesů
  - Definovat a vysvětlit distribuované algoritmy detekce zablokování

NSWI150 Virtualizace a cloud computing

- Vysvětlit motivaci pro virtualizaci, popsat taxonomie virtualizačních technologií.
- Popsat pojmy hardwarově asistovaná virtualizace, paravirtualizace, emulace. Popsat komunikaci mezi virtuálními stroji.
- Popsat podporu virtualizace na dnešních hardwarových architekturách a operačních systémech.
- Vysvětlit technologie kontejnerů, Linux namespaces, cgroups, docker, odlišné filosofie používání kontejnerů.
- Popsat různá hardwarová řešení pro datacentra.
- Popsat cluster, vyvažování zátěže, vysoká dostupnost, odolnost vůčí výpadkům.
- Popsat cloudové technologie, komponenty a služby. Vysvětlit termíny IaaS, PaaS a SaaS.
- Popsat exekuční modely – virtual machines, cloud services, containers, microservices, serverless computing, mobile services, high-performance computing. Popsat cloud management, orchestration, monitoring.
- Popsat metody správy, ukládání a zpracování dat, no-sql databáze v cloudu, map/reduce a související technologie.
- Popsat komunikaci a synchronizaci v cloudových platformách, CDN, caching.

NSWI131 Vyhodnocování výkonnosti počítačových systémů

- Měřící nástroje
  - Vysvětlit problémy spojené s režií (overhead) a rušením (perturbation) měření a ilustrovat je na měření času
  - Vysvětlit základní funkce hardware performance event counters (events, counters, PEBS, latency sampling)
  - Použít běžné nástroje (například perf, PAPI, VTune …) pro získávání informací z hardware performance event counters
  - Popsat základní funkce běžných mechanismů pro monitorování dalších metrik (například SNMP, JMX …)
  - Popsat mechanismus profilování programu, diskutovat jeho vlastnosti a omezení, použít základní profilovací nástroje (například perf) pro identifikaci výkonnostně citlivých míst programů
  - Popsat mechanismy instrumentace základních forem programu (zdrojový kód, bytecode, binární kód), diskutovat jejich omezení, použít základní instrumentační nástroje (alespoň jeden vybraný například z Coccinelle, AspectJ, DiSL, Pin, Valgrind, DynamoRio, bpftrace …) pro instrumentaci programů
- Měřící techniky
  - Popsat a použít základní techniky konstrukce experimentů (single factor, one factor at a time, n-fold validation)
- Výkonnostní metriky
  - Definovat běžné výkonnostní metriky (throughput, latency, jitter, availability, CPI/IPC, miss rate, MIPS, FLOPS, wall clock time, thread time) a diskutovat jejich vlastnosti (například přenositelnost či porovnatelnost)
  - Vysvětlit motivaci konstrukce výkonnostních benchmarků a uvést příklady benchmarků spolu s jejich metrikami (například SPEC CPU, SPEC JBB, TPC-C …)
- Zpracování měření
  - Definovat a popsat použití základních technik filtrování odlehlých pozorování (winsorizace)
  - Definovat a popsat použití základních statistik (průměr, medián, kvantily, rozptyl) na měřená data
  - Definovat intervaly spolehlivosti a použít parametrické metody a bootstrap pro výpočet intervalu spolehlivosti průměru měřených dat (není nutná znalost vzorců konkrétních výpočtů, ale očekává se schopnost takové vzorce v případě potřeby dohledat a použít)
  - Použít statistické testování hypotéz pro vyhodnocení měřených dat (není nutná znalost vzorců konkrétních výpočtů, ale očekává se schopnost takové vzorce v případě potřeby dohledat a použít)
  - Použít běžné typy grafů (scatterplot, histogram, Q-Q, box-and-whiskers, violins) pro vizualizaci výsledků měření

Pokryto předměty

- NSWI109 Konstrukce překladačů
- NPRG058 Pokročilé programování v paralelním prostředí
- NSWI035 Principy distribuovaných systémů
- NSWI150 Virtualizace a cloud computing
- NSWI131 Vyhodnocování výkonnosti počítačových systémů