

KOMUNIKAČNÍ A PROPRIETÁRNÍ KOMUNIKAČNÍ PROTOKOLY BEZPEČNOSTNÍCH TECHNOLOGIÍ

Communication and Proprietary Communication Protocol of
Security Technologies

Robert Miňovský

Bakalářská práce
2013



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2012/2013

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Robert MIŇOVSKÝ**
Osobní číslo: **A10753**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Bezpečnostní technologie, systémy a management**
Forma studia: **kombinovaná**

Téma práce: **Komunikační protokoly v bezpečnostních
informačních systémech**

Zásady pro vypracování:

- 1. Vytvořte literární rešerši na zadané téma.**
- 2. Vysvětlete význam komunikačních protokolů.**
- 3. Vypracujte teorii a zásady tvorby komunikačních protokolů využitím UML.**
- 4. Využijte vrstevnatou architekturu modelu komunikačních protokolů.**
- 5. Popište metodiku testování komunikačních protokolů.**
- 6. Vypracujte přehled využití proprietárních komunikačních protokolů a jejich využití v BIS.**
- 7. Vytvořte přehled stavu technické normalizace – požadavky platných norem na komunikaci BIS. Využití a uplatnění existujících standardů komunikačních protokolů v BIS, praktiky výrobců BIS technologií.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: tištěná/elektronická

Seznam odborné literatury:

1. POPOVIC, Miroslav. Communication protocol engineering. Boca Raton: Taylor, 2006, ISBN 9780849398148.
2. KRAVAL, Ilja, RNDr. Úvod do sémantiky UML 1.3 podle standardu OMG s komentářem. 19.8.2001. Lipina 100, 766 01 Valašské Klobouky: Object Consulting.
3. ITU-T Rec. X.200 (1994 E). Data Networks and Open system communication: Open system Interconnection – model and notation. 07.1994. Dostupné z: <http://www.itu.int/rec/T-REC-X.200-199407-I/en>
4. ČSN EN 50131, 50132, 50133, 50136. Soubor norem – poplachové systémy. Český normalizační institut.
5. EN 54. Soubor norem – Elektrická požární signalizace. Český normalizační institut.
6. PROTOCOL LAYERING. GREBE, David L. APOGEE LABS, Inc. [online]. [cit. 2012-12-15]. Dostupné z: <http://apogeelabs.com/pdffiles/Layering.PDF>

Vedoucí bakalářské práce:

doc. Ing. Martin Sysel, Ph.D.

Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce:

25. února 2013

Termín odevzdání bakalářské práce:

30. května 2013

Ve Zlíně dne 25. února 2013

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Milan Adámek, Ph.D.
ředitel ústavu

Příjmení a jméno: ROBERT MINOUSKY

Obor: RTSY

PROHLÁŠENÍ

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby¹⁾;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3²⁾;
- beru na vědomí, že podle § 60³⁾ odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60³⁾ odst. 2 a 3 mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považuji se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Ve Zlíně 27.3.2013

1) zákon č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, § 47 Zveřejňování závěrečných prací:

(1) Vysoká škola nevyjádřeně zveřejňuje disertační, diplomové, bakalářské a rigorózní práce, u kterých proběhla obhajoba, včetně posudků oponentů a výsledku obhajoby prostřednictvím databáze kvalifikačních prací, kterou spravuje. Způsob zveřejnění stanoví vnitřní předpis vysoké školy.

(2) Disertační, diplomové, bakalářské a rigorózní práce odevzdané uchazečem k obhajobě musí být též nejméně pět pracovních dnů před konáním obhajoby zveřejněny k nahlédnutí veřejnosti v místě určeném vnitřním předpisem vysoké školy nebo není-li tak určeno, v místě pracoviště vysoké školy, kde se má konat obhajoba práce. Každý si může ze zveřejněné práce pořizovat na své náklady výpisy, oplaty nebo rozmnoženiny.

(3) Platí, že odevzdáním práce autor souhlasí se zveřejněním své práce podle tohoto zákona, bez ohledu na výsledek obhajoby.

2) zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, § 35 odst. 3:

(3) Do práva autorského také nezahrnuje škola nebo školské či vzdělávací zařízení, užíje-li nikoli za účelem přímého nebo nepřímého hospodářského nebo obchodního prospěchu k výuce nebo k vlastní potřebě dílo vytvořené žákem nebo studentem ke splnění školních nebo studijních povinností vyplývajících z jeho právního vztahu ke škole nebo školskému či vzdělávacímu zařízení (školní dílo).

3) zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, § 60 Školní díla:

(1) Škola nebo školské či vzdělávací zařízení mají za obvyklých podmínek právo na uzavření licenční smlouvy o užití školního díla (§ 35 odst.

3). Odpírá-li autor takového díla udělit svolení bez vážného důvodu, mohou se tyto osoby domáhat nahrazení cizoběžného projevu jeho vůle u soudu. Ustanovení § 35 odst. 3 zůstává nedotčeno.

(2) Není-li sjednáno jinak, může autor školního díla své dílo užit či poskytnout jinému licenci, není-li to v rozporu s oprávněnými zájmy školy nebo školského či vzdělávacího zařízení.

(3) Škola nebo školské či vzdělávací zařízení jsou oprávněny požadovat, aby jim autor školního díla z výdělku jím dosaženého v souvislosti s užitím díla či poskytnutím licence podle odstavce 2 přiměřeně přispěl na úhradu nákladů, které na vytvoření díla vynaložily, a to podle okolností až do jejich skutečné výše; přitom se přikládá k výši výdělku dosaženého školou nebo školským či vzdělávacím zařízením z užití školního díla podle odstavce 1.

ABSTRAKT

Bakalářská práce přináší teoretický rozbor technik modelování a testování komunikačních protokolů se zaměřením do aplikace v bezpečnostních informačních systémech. Zkoumá vztahy mezi praktikami uplatňovanými v teorii konstrukce komunikačních protokolů a požadavky jejich praktické aplikace v technologiích bezpečnostních informačních systémů. Rozebírá aspekty návrhu vrstevnatých zásobníků komunikačních protokolů, jejich význam a uplatnění a dále analyzuje aspekty navrhování a aplikace proprietárních komunikačních protokolů.

Klíčová slova: Komunikační protokol, UML, bezpečnostní informační systémy, OSI, vrstevnatý model, testování komunikačních protokolů, proprietární komunikační protokol

ABSTRACT

This bachelor thesis provides theoretical analysis of modeling and testing techniques for communication protocols, focused to the application in the security systems. The thesis explores relationships between the practices applied in the design theory of communication protocols and requirements of their practical application in the security systems technologies. It also discusses design aspects of the communication protocol layered stacks, their importance and application. Furthermore it discusses design aspects of the proprietary communication protocols and their and usage.

Keywords: Communication protocol, UML, Security systems, OSI, layered protocol stack, communication protocol testing, proprietary communication protocol.

Na tomto místě si mnoho lidí zaslouží poděkování. Mezi těmi, které zde lze vyjmenovat je v první řadě má žena Svatava, která po 3 léta trpělivě vytvářela podmínky a atmosféru k tomu, že dnes mohu psát tyto řádky a náš malý syn, který ne vždy mohl dostat to, co právě chtěl.

Poděkování zaslouží všichni kantoři Univerzity Tomáše Bati, kteří nás poctivě učili, zejména pak Martin Sysel, který se ujal vedení této práce, přispěl cennými radami a umožnil vznik tohoto tématu. Sluší se hodně poděkovat kolegům vývojového oddělení JABLOTRONu, kteří mi poskytli cenné informace a pohledy do praktické části této práce a vedení JABLOTRONu, které podpořilo má studia časově. Dále to byli Zdeněk Chmelař, jehož názory mi byly cenou inspirací i pochopením věcí zapeklých a Tomáš Mikula, bez jehož nadhledu by pochopení některých norem bylo mnohem těžší.

Kromě toho by tato bakalářská práce nevznikla bez potřebné literatury a tvrdé práce všech jejích autorů. I jim patří mé poděkování.

Robert Miňovský.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická, nahraná do IS/STAG, jsou totožné.

OBSAH

OBSAH	3
ÚVOD.....	5
TEORETICKÁ ČÁST	6
1 TEORIE DIGITÁLNÍCH KOMUNIKAČNÍCH PROTOKOLŮ	7
1.1 ZÁKLADNÍ DEFINICE A VÝZNAM KOMUNIKAČNÍCH PROTOKOLŮ	7
1.2 SPOLEČNÉ ASPEKTY NAVRHOVÁNÍ KOMUNIKAČNÍCH PROTOKOLŮ	8
2 MODELOVÁNÍ KOMUNIKAČNÍCH PROTOKOLŮ	11
2.1 ANALÝZA	11
2.2 UNIFIED MODELING LANGUAGE (UML)	11
2.3 APLIKACE UML V MODELOVÁNÍ KOMUNIKAČNÍCH PROTOKOLŮ	12
2.4 UML OBJEKTY MODELOVÁNÍ KOMUNIKAČNÍCH PROTOKOLŮ	13
2.5 STAVOVÝ STROJ, STAVY	15
2.6 UML DIAGRAMY MODELOVÁNÍ KOMUNIKAČNÍCH PROTOKOLŮ	19
3 VRSTEVNATÁ ARCHITEKTURA KOMUNIKAČNÍCH MODELŮ	43
3.1 VÝZNAM VRSTEVNATOSTI KOMUNIKAČNÍHO MODLU	43
3.2 MODEL OTEVŘENÉHO PROPOJITELNÉHO SYSTÉMU (OSI)	44
3.3 VLIV VRSTEVNATÉ ARCHITEKTURY NA KONSTRUKCI PROTOKOLŮ	44
3.4 REFERENČNÍ MODEL VRSTEVNATÉ ARCHITEKTURY DLE STANDARDU X.200	44
3.5 OBECNÉ ZÁSADY PŘI URČOVÁNÍ VRSTEV	51
4 TESTOVÁNÍ KOMUNIKAČNÍCH PROTOKOLŮ	52
4.1 ZÁKLADNÍ PŘEDPOKLADY	52
4.2 TESTOVÁNÍ PODLE KONVENČNÍHO SW INŽENÝRSTVÍ	53
4.3 PŘÍSTUPY CLEANROOM METODOLOGIE	56
4.4 TEST FRAMEWORK	63
PRAKTICKÁ ČÁST	64
1 PROPRIETÁRNÍ A VEŘEJNÉ KOMUNIKAČNÍ PROTOKOLY	65
1.1 UŽITÍ PROPRIETÁRNÍCH KOMUNIKAČNÍCH PROTOKOLŮ	65
1.2 VÝVOJ PROPRIETÁRNÍCH KOMUNIKAČNÍCH PROTOKOLŮ	66
2 POŽADAVKY NOREM NA KOMUNIKACI BEZPEČNOSTNÍCH TECHNOLOGIÍ	71
2.1 I&HAS	71
2.2 SYSTÉMY KONTROLY VSTUPU (SKV)	73
2.3 SYSTÉMY PRŮMYSLOVÉ TELEVIZE (CCTV)	74
2.4 SYSTÉMY ELEKTRICKÉ POŽÁRNÍ SIGNALIZACE (EPS)	75
2.5 SYSTÉMY PŘIVOLÁNÍ POMOCI	76
2.6 SYSTÉMY DOHLEDOVÉHO A PŘÍJÍMACÍHO POPLACHOVÉHO CENTRA (MARC)	77
2.7 SHRUTÍ	78
3 VRSTEVNATOST V PRAXI BIS	80
4 PRAKTIKY KONSTRUKCE PKP VÝROBCŮ BIS:	81

4.1	ZÁKLADNÍ KONCEPCE	81
4.2	TECHNIKY MODELOVÁNÍ A VÝVOJE:	82
5	KOMUNIKAČNÍ PROTOKOLY V PRAXI BIS	84
	ZÁVĚR	85
	SEZNAM POUŽITÉ LITERATURY.....	87
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	90
	SEZNAM OBRÁZKŮ	91
	SEZNAM TABULEK.....	92
	SEZNAM PŘÍLOH.....	93

ÚVOD

Význam bezpečnostních technologií neustále narůstá. Tento trend ovlivňuje řada pozitivních faktorů. Z technologických především rostoucí účinnost, funkcionalita, integrace a masová výroba elektroniky a její klesající náklady, a to vč. vývojových. Dále je to rozvoj detekčních technologií a principů, které umožňují získávat stále levněji stále přesnější informace o sledovaných jevech a událostech automatickou cestou a v řadě případů na ně i automaticky reagovat ať zčásti nebo celkově.

Z netechnických faktorů jde především o životní úroveň, se kterou stoupá význam hodnot, které nelze vyjádřit exaktně a jsou proto často obtížně nebo vůbec pojistitelné. Kromě prevence škod, které nelze exaktně modelovat, tedy ani vyjádřit a prokázat, jde i o pocity jistoty, důvěry, bezpečí, soukromí a klidu.

Ve světě, který stále zvyšuje tempo a klade stále vyšší požadavky na výkony a angažovanost každé entity, vzniká potřeba důvěryhodně udržet tyto jistoty při vynaložení minimálního úsilí, aby se okruh nároků a kontrolovaných oblastí mohl snížit a pozornost bylo možné věnovat klíčovým tématům.

Bakalářská práce se soustředí na obecnou problematiku digitálních komunikačních protokolů, jejich aplikaci a využití v technických systémech komerční a bezpečnosti. Proto, že zvládání rizik nutně začíná sběrem a předáváním dat a vyhodnocením a signalizací informace je toto téma nanejvýš aktuální. V tomto oboru nadále roste potřeba integrace dříve ryze bezpečnostních systémů do sofistikovaných celků umělé inteligence a automatizace, protože právě tato cesta příznivě ovlivňuje finální investiční a provozní náklady systémů jako celku, jejich efektivitu a spolehlivost. Aktuálním tématem je a bude stále vyšší distribuovanost systémů, jejich dálkový dohled a z toho plynoucí otázky na zajištění odpovídajících parametrů standardu komunikací na různých úrovních technologií. Je zřejmé, že soudobá generace bezpečnostních technologií vykračuje z uzavřených signalizačních standardů směrem k propojování a distribuci systémů a tím bude dále přebírat techniky známé z prostředí Internetu a průmyslové automatizace.

I. TEORETICKÁ ČÁST

1 TEORIE DIGITÁLNÍCH KOMUNIKAČNÍCH PROTOKOLŮ

Tato část popisuje obecnou teorii **návrhu** digitálních komunikačních protokolů. Vzhledem k rozsahu teorie tohoto tématu byla zvolena technika modelování využitím UML (Unified Modeling Language). Ukazuje praktické využití teoretického modelovacího jazyka a dále ji analogicky lze využít v další praxi oboru bezpečnostních technologií – např. k modelování procesů, logického chování aplikovaných systémů apod.

1.1 Základní definice a význam komunikačních protokolů

Existuje řada teoretických definic termínu ‚komunikační protokol‘ (dále též jen *protokol*). Ty jsou vždy zabarveny účelem a částí širokého celku tohoto tématu. Pro účely této práce bude použita definice složená z případů uvedených v [1] kap. 1.1:

- Soubor pravidel, podle kterých dvě geograficky oddělené komunikační entity zpracovávají zprávy, které si vyměňují.
- Zpracováním se míní kontrola technických parametrů, formátu, případně správnosti a obsahu zasílaných zpráv.
- Geografická oddělenost, resp. různé komunikační entity představují především běh těchto entit na různých procesorech. Podle konkrétní konstrukce lze tuto definici aplikovat přiměřeně s ohledem na to, zda jsou komunikující entity (zařízení) fyzicky oddělené, resp. propojená komunikačním kanálem.

Obecně je komunikační protokol metodika, jak přenášet informace **konkrétního typu** mezi dvěma zařízeními, které si je potřebují vyměňovat. Typ komunikované informace představuje účel, kterému komunikace slouží. Z hlediska obecné aplikace v bezpečnostních informačních systémech jde především o služby:

Synchronizace	Přenos dat (bez rozdílu druhu) s cílem mít shodné kopie (v reálném čase) na více místech
Signalizace	Přenos dat z místa vzniku a detekce události (jevu) o této události (jevu) na místo jejich sběru a zpracování.
Povelování, řízení	Zasílání dat nesoucích informací pro výkon určitých instrukcí a/nebo o zpětné vazbě o jejich vykonání
Selektivní přenos	Přenos vybrané části dat na základě instrukce s cílem redukce objemu a k dalšímu zpracování v místě příjmu.

Digitální komunikační protokoly jsou zpracovávány digitálními technologiemi, jejichž neodmyslitelnou součástí je procesor jako entita zpracování informace nesené signálem. Procesory jsou využívány aplikačním vybavením daného zařízení, resp. jeho **procesy**, které využívají *protokoly* k předávání dat. [1] kap. 1.1

Proces je definován jako: jednotka (soubor vykonatelných instrukcí), kterou je možno nechat přiřadit procesoru a nechat ji vykonat procesorem, včetně **aktuálních proměnných svého stavu a prostředků přiřazených výpočetním zařízením**. [2], kap. 3.1. Tato definice má následnou přímou souvislost s přípravou testovacích scénářů, viz kapitola 4.

Produktem procesu využívajícího *protokol* k zasílání zpráv jsou datové jednotky (DU), nazývané podle účelu a druhu učení *zprávy* (messages) nebo *pakety* (packets), *rámcy* (frames), *datagramy* (datagrams) apod. Použití konkrétního významu je obvykle příslušné účelu daného *protokolu*, konvenci a struktury nesených dat. Rovněž se liší podle vrstvy ISO modelu, na které daný *protokol* pracuje. Viz kap. 3. [11]

Podle definice procesu (viz 1.1) **zpráva** představuje data uvedená procesem do očekávaného formátu (tvaru) podle konvencí daného komunikačního protokolu.



Obr. 1: Obecný postup aplikace komunikačního protokolu

Komunikační proces, jako způsob předávání dat mezi oddělenými výpočetními entitami lze popsat komunikačním protokolem [1], kap. 1.1.

1.2 Společné aspekty navrhování komunikačních protokolů

Jako existuje řada teoretických definic pojmu komunikační protokol, existují i různé konkrétní přístupy k jejich konstrukci. Inženýring *protokolů* se vyvinul jako technická disciplína na základě potřeb praxe – jednotlivé části tohoto oboru pokrývají svým aparátem požadavky spotřebitelů komunikačních protokolů.

I přes existující rozdíly lze vybrat skupinu vlastností, které jsou společné všem přístupům, které jsou popsány v následujících kapitolách.

1.2.1 FORMÁT ZPRÁV (MESSAGE FORMAT)

Definuje strukturu (délku, pořadí, počet polí, separátory, konstanty, iniciační hodnoty a kódovací schéma (BIN, ASCII, apod.)); určuje jak se data do/ze zprávy zapisují a čtou.

Zpráva typicky začíná hlavičkou, která blíže specifikuje parametry a charakter dat ve zprávě nesených, podružné položky a informace pro řízení komunikace (výměnu zpráv).

0	4	8	16	19	31
Version	IHL	Type of Service	Total Length		
Identification			Flags	Fragment Offset	
Time To Live		Protocol	Header Checksum		
Source IP Address					
Destination IP Address					
Options					Padding

Obr. 2: Příklad formátu zasílané zprávy - IP paket v.4.

(Zdroj:[2])

1.2.2 ZPRACOVÁNÍ ZPRÁV (MESSAGE-PROCESSING)

Soustava komunikačních funkcí (operací), které jsou provedeny s přijatou (odesílanou) zprávou. Jsou nedílnou součástí definice *protokolu*, resp. funkcí, které *protokol* plní na odpovídající vrstvě vícevrstvého komunikačního modelu – viz kapitola 3.

1.2.3 ZPRACOVÁNÍ CHYB (ERROR-PROCESSING)

Soustava funkcí, které vykoná obslužná rutina *protokolu* v případě, že nastane očekávaná nebo neočekávaná chyba v komunikačním procesu (např. detekce poškozených dat ve zprávě, přeházené pořadí doručených zpráv, ztracená zpráva, přetečení časovačů apod.).

Rozdíl mezi očekávanou a neočekávanou chybou je v tom že řadu očekávaných chyb ošetří korekční rutiny a mechanismy, které (rovněž) mohou být součástí modulu zpracování chyb.

1.2.4 FORMÁLNÍ SPECIFIKACE KOMUNIKAČNÍHO PROTOKOLU

K přesné specifikaci *protokolu* je nezbytný formálně přesný zápis, který definuje nejen jednotlivé rutiny obsluhy *protokolu* a jeho datové struktury, ale rovněž přesné pořadí, ve kterém jsou události a obslužné rutiny *protokolu* zpracovávány a na základě jakých událostí k tomu dochází.

K tomu slouží tzv. **formální definice komunikačního protokolu**. *Protokol* je modelován k tomu vhodnou technikou – jako **konečný automat** (KA; *Finite State Machine*; viz kap. 2.5.) tak, že všechny možné změny a reakce mají své přesné pořadí a podmínky, za kterých nastávají (jsou prováděny), včetně definice **chování v přechodových stavech**.

Proti tomu metoda neformální definice *protokolů* je textově orientovaná. Nepřináší jednoznačnou definici, není schopna exaktně popsat chování jednotlivých stavů a událostí v čase, jejich přechodové děje, větvení a vazby. Její **výsledky jsou nejednoznačné**. Implementace takto definovaného *protokolu* může v různých verzích přinášet nekompatibilitu, resp. neočekávané chování ([1], kap.2).

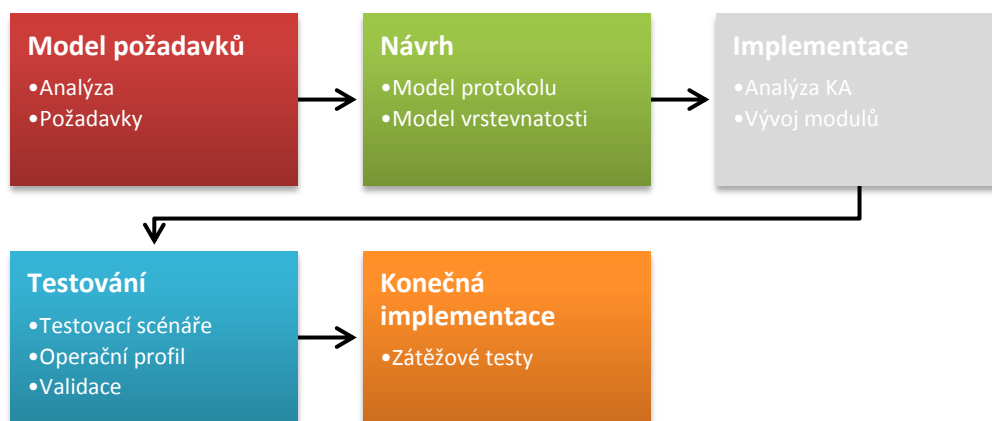
1.2.5 TESTOVÁNÍ

Je nezbytnou součástí vývoje *protokolů*. Má představovat sofistikovaný, exaktní a personálně nezávislý proces, zkoumající odpověď na otázku: „jak výsledek funguje“ – tedy zda bylo dosaženo definovaných konstrukčních předpokladů. Odhaluje a eliminuje chyby a jeho výstup představuje záruku použitelnosti předmětu vývoje. Dále zkoumá výkonnost navrženého *protokolu* a jeho použitelnost v zamýšlené aplikaci.

1.2.6 IMPLEMENTACE

Představuje sadu technik a nástrojů, kterými jsou návrh – formální specifikace *protokolu* – a následná zjištění z testů, přesně přeneseny do zdrojového kódu programovacího jazyka (C, C++, Java apod.). [1], kap. 2. Tak se, vývojový proces, dle Obr. 3, cyklicky opakuje.

Vzhledem k rozsahu se tato práce rozbořem implementačních technik *protokolů* nezabývá. Jejím cílem je vysvětlit techniky a metody návrhu a testování *protokolů*, které jsou v aplikační praxi důležité k pochopení a kontrole správné činnosti technologií.



Obr. 3: fáze konstrukce komunikačního protokolu

2 MODELOVÁNÍ KOMUNIKAČNÍCH PROTOKOLŮ

Kapitola rozebírá využití jazyka UML k modelování a formální návrh komunikačních protokolů. Pokud není uvedeno jinak, vychází tato kapitola z [1] – kapitoly 2 a 3. Představuje všechny potřebné typy UML diagramů a jejich objekty.

2.1 Analýza

Je úvodní obecně-projektovou fází návrhu systémů – v tomto případě *protokolů* a definuje:

- co je třeba udělat (jak to udělat) a
- jak výsledek ověřit.

Jejím výstupem jsou:

- a) **funkční požadavky**: definice chování *protokolu* (systému)
- b) **nefunkčními požadavky** důležité pro návrh: zpravidla obecné atributy, jako časové a technologické omezení, výkon, rozsah implementace atd.
- c) Množina uspořádaných **dvojic modelových případů**: každý požadavek (parametr, atribut) funkční analýzy *protokolu* na vstupu a metrika jeho zkušebního testu (očekávaného chování definovaného požadavku) na výstupu.

Tímto postupem se vytváří architektura *protokolu* (systému): Zkušební testy jsou odvozeny z jednotlivých funkčních požadavků; Jestliže konstruktér přemýšlí o požadovaném chování systému a způsobu jeho ověření, nevyhnutelně musí přemýšlet nad celkovou konstrukcí, tzn., definice architektury zpřesňuje funkční požadavky a opačně. *Zdroj: [1], kap. 2.*

2.2 Unified Modeling Language (UML)

Jazyk pro specifikaci, vizualizaci, konstrukci a dokumentaci produktů v oblasti objektově orientovaného programování a modelování procesů (objektů). Zavádí objektově orientovaný přístup pomocí **abstrakce** (koncentrace na určitý pohled, při vynechání (zanedbání) prvků modelu, které nejsou pro daný pohled relevantní. Je nezávislý na jiných programovacích jazycích, není programovacím jazykem. Ale v důsledku může zdrojový kód nebo jeho části vytvářet.

Předmětem modelování není informační systém, ale procesy objektů. Jde o analýzu prostředí (podmínek), kde bude informační systém nasazen. Provádí se také z důvodů optimalizace procesů, jejich základního popisu a **odhalení chyb**.

Základní techniky a přínosy abstrakce jsou:

- zjednodušují a zefektivňují modelování
- systém je modelován několika druhy pohledů – modelů
- Každý model je vyjádřen s větší nebo menší mírou detailů (zoom), které odkrývají dekompozici, resp. skrývají hlubší detaily
- Jako standardní úrovně abstrakce se uvádí:
 - strategické modelování (úvodní fáze projektu),
 - model analýzy (model nezávislý na prostředí),
 - model designu (model pro dané prostředí),
 - model implementace (do daného prostředí s implementačními detaily)
 - kódování.

V neposlední řadě UML představuje standard, což je vždy důležité z hlediska kompatibility a spolehlivosti používaných prostředků obecně. V ČR je standard jazyka přijat v podobě technické směrnice ČSN P ISO/TS 19103. *Zdroj kapitoly: [4], str. 10 až 17*

2.3 Aplikace UML v modelování komunikačních protokolů

Panuje shoda akademického prostředí a průmyslu na použití UML při konstrukci *protokolů*. Nepředstavuje však jedinou správnou cestu. Další techniky modelování *protokolů* představuje např.:

- SDL (Specification and Description Language) jako standard ITU-T Z.100
- MSC (Message Sequence Charts Language) jako standard ITU-T Z.120
- [7] spatřuje jako lepší transformaci funkčních požadavků *protokolu* do modelu chování v podobě KA.
- Dále jsou to kombinace uvedených technik, např. v podobě doporučení ITU-T Z.109, které standardizuje UML profil pro aplikaci při modelování v SDL.

UML využitím grafického popisu (na rozdíl od neformální textové definice) umožňuje vyjádřit klíčové definice celíku funkčních požadavků *protokolu*, jako jsou jednotlivé stavy, **jejich přechody, vazby objektů atd.** Tím významně přispívá ke správnosti a efektivitě návrhu. *Zdroj kapitoly: [1], kap. 3*

2.4 UML objekty modelování komunikačních protokolů

Chování *protokolu* modeluje série **diagramů** uvedených dále. Poskytují zvolenou míru abstrakce a její druhy – viz [1], kap. 2.2. České názvosloví je převzato z [4] a u některých diagramů se mění podle verze UML (1.x a 2.0). Seznam diagramů uvádí [1], kap. 1.

Tabulka 1: Přehled UML diagramů pro modelování komunikačních protokolů

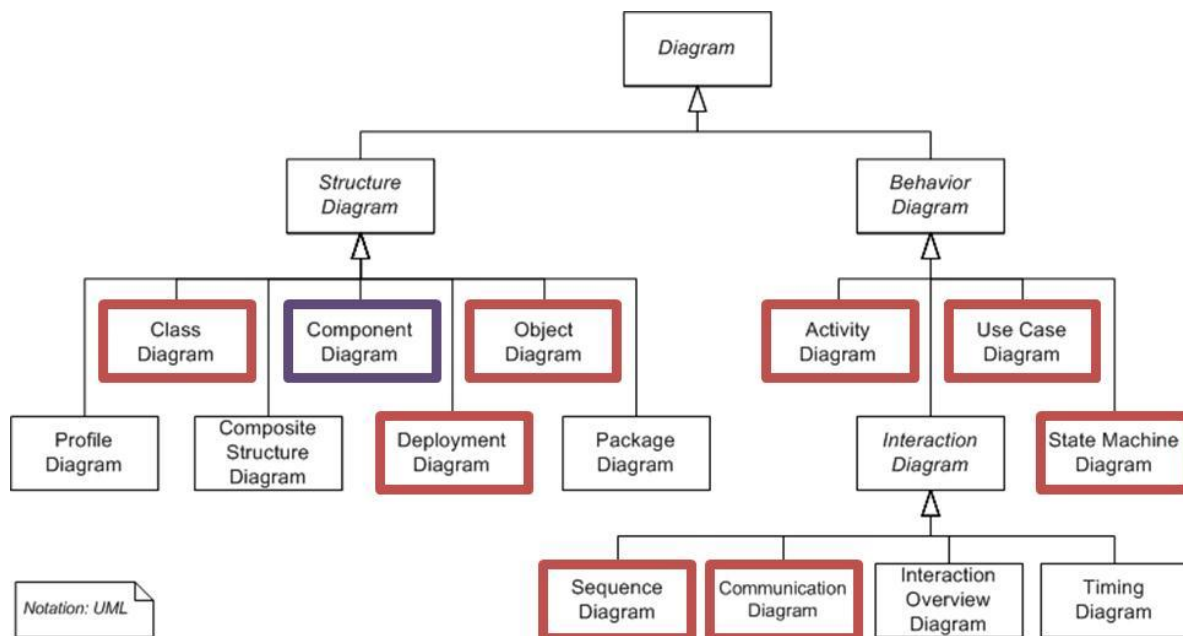
Typ diagramu		Použití
Fáze požadavků a analýzy		
1.	Případ užití (Use case)	Dynamické aspekty chování
2.	Diagram spolupráce (Collaboration diagram); též Communication Diagram - UML 2.0)	Dynamické aspekty chování, interakce
Fáze návrhu		
3.	Diagram tříd (Class Diagram)	Statické aspekty
4.	Objektový diagram (Object Diagram); též diagram instancí	Statické aspekty
5.	Sekvenční diagram (Sequence Diagram)	Dynamické aspekty, interakce
6.	Diagram aktivit (Activity Diagram)	Vnitřní tok událostí a řízení
7.	Stavový diagram (Statechart Diagram); též State Machine Diagram - UML 2.0)	Životní cyklus sekvencí instancí
8.	Diagram rozmístění zdrojů (Deployment Diagram)	Fyzická topologie

Kapitola vysvětluje význam UML diagramů v procesu modelování *protokolů*. Neřeší 100% detailů parametrů a podmínek aplikace všech *prvků*, ani jejich detailní seznam. Tyto informace jsou dostupné např. v [1], [5], [6] nebo na adrese:

http://www.omg.org/news/meetings/workshops/presentations/uml_presentations/

Na Obr. 4 je struktura diagramů UML. Červeným rámem jsou označeny ty, které přímo popisují model chování *protokolu*, modrým je – jako doplněk – zvýrazněn *komponentový diagram*, který se užívá ve fázi implementace modelovaného návrhu.

Z obrázku je zřejmé, že model UML popisuje dva základní aspekty modelovaného *protokolu*, totiž: **CHOVÁNÍ** a **STRUKTURU**, které společně vytváří **UML diagram**.



Obr. 4: Struktura diagramů modelování protokolů v modelu UML

Zdroj: [3]

2.4.1 VYBRANÉ VLASTNOSTI PRVKŮ A DIAGRAMŮ UML

Všechny *prvky UML modelu* mají vlastnosti – parametry, které uchovávají nebo určují hodnoty důležité pro specifické chování každého *prvku*. Modelování probíhá umísťováním prvků do diagramu a nastavováním jejich vlastností.

Diagramy a prvky v tomto dokumentu jsou modelovány za použití nástroje StarUML [6], který byl zvolen z důvodu vyspělého přístupu. Neprodukuje “jen” obrázky, ale vytváří **kompletní model**, popsáný vlastnostmi jednotlivých *prvků*. Rovněž je schopen generovat části implementačních kódů *modelu* v jazycích C#, C++ a JAVA.

Mezi společné vlastnosti všech *prvků* a *relací* je třeba uvést **omezení (constraints)**, **označené hodnoty (tagged values)** a **stereotyp (stereotype)**. Všechny **zavádí možnost přidávat uživatelsky definované informace** do *modelu*, bez narušení sémantiky UML.

Omezení (Constraints): potomek *ModelElement*; vyjadřuje podmínku, nutnou pro správné fungování *modelu*, která je obsažena jako výraz v jeho atributu *body*. Může být přiřazeno k *prvku* modelu a ke *stereotypu*. *Omezení* přiřazené *stereotypu* se promítne ke všem *prvkům* *modelu*, kterým je přiřazen daný *stereotyp*.

Označená hodnota (*Tagged value*): představuje vlastnost vyjádřenou dvojicí (význam – hodnota). Je jako možnost přidat k libovolnému *prvku modelu* svou uživatelsky definovanou vlastnost, přiřazena buď jako *kompozice* k *prvku modelu* nebo ke *stereotypu* (ale ne současně).

Stereotyp (*stereotype*): mechanismus klasifikace *prvků modelu* uživatelskými kategoriemi. V *modelu* zavedená kategorie, do které lze zařadit *prvek modelu*. Může definovat novou kategorii *prvků modelu*. Ty mají stejnou strukturu a chování jako nekategorizované *prvky*, ale mají jiný význam. Lze použít standardní *stereotypy* definované UML. Vztah mezi *prvkem modelu* a *stereotypem* je běžnou *asociací*. Daná *instance prvku* může mít přiřazen max. jeden *stereotyp*. *Prvek*, přebírá také *označené hodnoty* daného *stereotypu*. Existují tedy dvě možnosti, jak přiřadit *prvku modelu* *označenou hodnotu*.

Informace o dalších, konkrétních, vlastnostech a attributech *prvků* jsou uvedeny v tabulkách *prvků* jednotlivých diagramů dále a v PŘÍLOHA 1. Zdroj: [4], kap. 3.4.

2.4.2 VYBRANÉ VLASTNOSTI RELACÍ

Relace (*Relationships*) vyjadřují vztahy mezi *prvky*. Existují tři základní typy *relací*: {*Generalizace*, *Asociace*, *Tok*}. PŘÍLOHA 2 uvádí *relace* používané v *diagramech* modelování *protokolů* a jejich důležité vlastnosti. Většina z nich je potomkem třídy *Relace*.

2.5 Stavový stroj, stavy

Modelování *protokolů* je založeno na teorii KA. Tato kapitola popisuje základní aspekty *Stavového stroje* UML, který definuje typy *stavů* a jejich vztahy.

UML zavádí *svazek stavové stroje* (*package state machines*), který je *sub-svazkem behavioral package*. Modeluje chování pomocí KA a UML je dále rozpracovává do objektově orientovaného prostředí. [4]

Základní předpoklad *stavového stroje* je, že libovolný *prvek* může obsahovat {N} *stavových strojů*, které modelují jeho chování. Metatřída *stavový stroj* (*StateMachine*) je potomek *prvku modelu* a obsahuje jako *kompozici* (jsou jen jeho) *vrcholové stavy* (hierarchicky nejvyšší *stav stavového stroje*) a *přechody* (*transitions*). [4]

Podle UML terminologie představuje [1]:

- **konečný automat:** posloupnost stavů, jimiž objekt prochází v jeho životním cyklu,
- **stav:** objekt splňuje určité podmínky, koná činnost nebo čeká na událost,

- **událost:** výskyt podnětu, který spouští přechod stavu (objektu),
- **akce:** nepřerušitelná (výpočetní) operace systému,
- **aktivita:** dělitelný sled akcí a dalších aktivit.
- **přechod (stavu):** vztah mezi zdrojovým a cílovým stavem (dva různé stavy nebo jeden stav), který specifikuje akce, jež mají být provedeny, když nastane daná událost a daná ochranná podmínka je splněna.

Klíčové abstrakce ohledně KA jsou [1]:

- **stav objektu:** část životního cyklu objektu (daná určitou podmínkou, doba určité činnosti nebo interval, ve kterém objekt čeká na určité události).
- **změna stavu:** krátký intervalu životního cyklu objektu, který se vztahuje k akcím vyvolaným určitou událostí. Specifikuje ho pět atributů: *{výchozí stav, spouštěč události, ochranná podmínka, akce, cílový stav}*.

Dále jsou uvedeny UML *objekty*, které souvisí s popisem a definicí stavů obecně:

2.5.1 STAV (STATE):

- Situace, kdy se definované podmínky nemění. Obecně entita čekající na *událost*, nebo probíhající *akce*, udržující entitu v daném *stavu* až do ukončení této aktivity.
- Může mít své vnitřní přechody (*InternalTransition*), které nemění daný *stav*. Zdroj a cíl těchto přechodů jsou totožné *stavy*.
- Potomek *stavového vrcholu* (*StateVertex*), přebírá jeho vlastnosti, vč. možnosti vstupovat do interakce *přechodu* mezi *stavy*.
- Mezi *stavovým vrcholem* (tedy i *stavem*) a *přechodem* jsou dvě asociace, které vyjadřují, že *přechod* mezi *stavovými vrcholy* je směrový – rozlišuje zdrojový (source) a cílový (target) *stavový vrchol* a spojuje je.
- Může mít *{0..N}* *pozdržených událostí* (*defferableEvents*), které jsou vyvolány, ale nejsou zpracovány. Zpracovány budou v jiném (určeném) *stavu*; viz kap. 2.5.4.

Stav může vykonávat *akce*. Každý *prvek třídy stav* má kolekci *{0..1}* resp. *{0..N}* *akcí* (podle konkrétního typu *objektu* a implementace) a dělí se na dále uvedené typy:

- **Vstupní Akce (EntryAction):** spustí se, když entita vstoupí do daného *stavu*, nezávisle na tom z jakého jiného *stavu* se do daného dostala.
- **Výstupní Akce (ExitAction):** spustí se, když entita daný *stav* opouští, nezávisle na tom, do kterého *stavu* přechází.

- **Vnitřní Akce (DoActivity):** běží, pokud je entita v daném *stavu*. Představují *aktivní stabilní stavy* (provádějí nějakou činnost v době, kdy je *objekt* blokován a čeká na *událost*) nebo *krátkodobé stavy* (vykonají určité výpočty a skončí).

Používá se osm základních typů akcí [1]:

- Vytvoření objektu
- Ukončení objektu
- Volání operace jiného objektu
- Volání operace vlastního objektu (lokální invokace)
- Odeslání signálu (zprávy) jinému nebo vlastnímu objektu
- Zpracování návratové hodnoty
- Provedení atomické operace
- Ukončení provádění (operací)

Stav je abstraktní metatřída a do modelů vstupují její potomci:

- **Jednoduchý stav (SimpleState):** nemá žádné *sub-stavy*. Aktivita atomického charakteru (nedělitelná operace), *přechod* do dalšího *stavu* znamená její ukončení.
- **Konečný stav (FinalState):** nesmí mít žádné odchozí *přechody*
- **Kompozitní stav (CompositeState):** má *kompozici stavových vrcholů* jiných *sub-stavů*. *Kompozitní stavový vrchol* výlučně vlastní daný *kompozitní stav*. Vzniká spojením více *akčních stavů* a dalších *stavů* aktivity. Může zahrnovat speciální akce, např. obsluhu vstupu a výstupu. [1]
- **Pseudostav (Pseudostate):** vstupuje do modelů v různých formách – viz dále

2.5.2 PSEUDOSTAV (PSEUDOSTATE)

Abstrakce vrcholů grafu *stavového stroje* k přehlednému vyjádření složitých modelů. Má různé typy, které se rozlišují hodnotou atributu *PseudostateKind*:

- **Inicializační (Initial):** označuje začátek grafu – *přechod* do implicitního *stavu kompozitního stavu* (default state). *Kompozitní stav* má jeden *inicializační stav*.
- **Hluboká historie (Deephistory):** *stavový vrchol*, reprezentuje *stav* mezi *sub-stavy kompozitního stavu*, který byl aktivní, když systém daný *kompozitní stav* opouštěl. V modelu se zavede mezi *sub-stavy* tento vrchol a provede se *přechod* do něj. Pak

se tok řízení vrátí zpět do *sub-stavu kompozitního stavu*. *Kompozitní stav* může mít max. jeden *pseudostav Deephistory*.

- **Mělká historie (Shallowhistory):** *vrchol*, který reprezentuje minulý *stav*.
- **Spojení (Join):** spojení více vstupních *přechodů* do jednoho. Má {2..N} vstupů a jeden výstup. Vstupní *přechody* nesmí při *join* mít *ochrannou podmínku*.
- **Vidlička (Fork):** opak *join*. Jeden *přechod* se rozdělí do několika. Odchozí *přechody* nesmí mít *ochrannou podmínku*.
- **Výběr (Choice):** dynamicky hodnotí *ochranné podmínky přechodů* vycházející z jednoho vstupního toku. Dle výsledku dojde k výběru *přechodu*.
- **Synchronizační stav (SynchState):** synchronizuje různé části *stavového stroje*, které jsou v souběhu. Používá se s *fork* a *join*, zabezpečuje, že *stavy* opustí jednu část *stroje*, než vstoupí do druhé.

2.5.3 PŘECHOD (TRANSITION)

Potomek *prvku modelu (ModelElement)*. Směřovaný vztah mezi zdrojovým a cílovým *stavovým vrcholem (StateVertex)*. Obsahuje {0..1} *spouštěč (trigger)*. Může mít *ochrannou podmínku (guard)* – viz kap. 2.5.4. Obvykle probíhá až po skončení poslední činnosti ve výchozím stavu objektu.

2.5.4 UDÁLOST (EVENT):

Jev relevantní v modelování zkoumaného systému. Potomek *prvku modelu*. Sestává z:

- *Typu událost* (nese její vlastnosti – představuje „klasifikátor“ instance události)
- *Instance událost* (to, co se konkrétně vyskytne v daném okamžiku)

Plní funkci *spouštěče pro přechod*, tj. vyvolává *přechod*. K *přechodu* dojde jestliže:

- Zdrojový *stav* je aktivní
- Nastane *událost (spouštěč daného přechodu)*
- *Ochranná podmínka daného přechodu*, která se vyhodnocuje při výskytu *události spouštěče*, je *TRUE*.

Zvláštní případ představuje *událost*, vyvolána v průběhu nějaké *aktivity* v nějakém *stavu*. Tento *stav* na ni nereaguje a normálně je ztracena. Má-li být zachycena k zpracování v jiných (nebo opakovaných) *stavech*, musí se vyvolat jako *pozdržená událost (deferrable)*

event). Pak se *událost* pouze řadí, nezpracovává se a je připravená ke zpracování v příštích cyklech. Až se *stroj* dostane *stavu*, který nedrží tyto *události* jako pozdržené, zpracují se.

V UML existují různí potomci *Události*, kteří představují specializaci [1], [4]:

- **Událost volání (CallEvent):** přijetí požadavku synchronního volání specifické Operace (která způsobí vznik *události volání*). Dva speciální případy *události volání* jsou událost tvorby (*create*) a destrukce (*destroy*) objektu. Operace může být volána vnějším *objektem (instancí)* nebo sám sebou (*self*).
- **Událost signálu (SignalEvent):** přijetí asynchronního *signálu* (*signál* je vždy asynchronní) zasláného vnějším *objektem (instancí)* nebo sám sebou (*self*). Je ve vztahu *asociace* vůči *signálu*, který je přijímán. Modeluje se *stereotypem* <<signál>> nebo *relací stereotypu* <<send>>, mezi operací *třídy*, která *signál* posílá a *třídou*, která definuje jeho zdroj.
- **Časová událost (TimeEvent):** obecně reprezentuje klíčové slovo *after*, modeluje uplynutí určité časové (relativní anebo absolutní) lhůty. Čas vzniku a příjmu (zpracování) *události* se považuje za stejný, pokud je není třeba odlišit z technologických důvodů (distribuce, frontování apod.). Implicitně se časovač spouští *přechodem* z/do určitého *stavu*. Umožňuje implicitní time management.
- **Událost změny (ChangeEvent):** asynchronní *událost*, nejčastěji používaná. Obsahuje logický výraz (většinou ohledně hodnot *atributů*, naplnění vazeb apod.), který přechodem do stavu *TRUE*, vyvolá *ChangeEvent*. Obecně představuje význam klíčového slova *when*, ale **není** *ochrannou podmínkou*!

Zdroj kap. 2.5: [4], pokud v textu není konkretizováno.

2.6 UML diagramy modelování komunikačních protokolů

2.6.1 PŘÍPAD UŽITÍ (USE CASE)



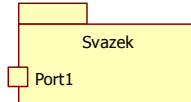
Diagramy *případů užití* jsou výchozí bod vývoje systémů. Efektivně nahrazuje chaotické dotazy a textového sepisování požadavků přesnou sémantiku [4] – grafickým popisem.

Ve **Fázi definice požadavků** každý *prvek případ užití* zachycuje jeden funkční požadavek modelovaného systému, doplněný odpovídajícími nefunkcionálními požadavky. Diagram popisuje úzce související funkcionalitu (sekvenci událostí, interakci) množiny objektů (typicky stejné *třídy* – nejmenší logickou jednotku, pro kterou lze vytvořit *případ užití* SW

modelu [4]). Modeluje soustavu specifikací, služeb, které bude systém poskytovat a jejich rozhraní a pomáhá tyto objekty hierarchicky seskupit a pracovat s nimi. Neřeší, jak budou tyto požadavky splněny.

Množina *případů užití* popisuje chování subsystému (*třídy* nebo *rozhraní*) a jeho interakce s okolím [4]. Představuje model požadavků, který zachycuje chování a další, technické a procesní nároky a požadavky. Používá se ve všech dalších fázích modelu, zejména analytické části, návrhu, implementaci, vývoj testovacích sad, až po nasazení.

Tabulka 2: Základní objekty diagramu případu užití

#	Grafický symbol	Popis
1.	 <p>Aktor</p> <p>Aktér (Actor)</p>	<p>Potomek: <i>Klasifikátor</i></p> <p>Množina logických rolí (člověk, stroj, SW komponenty) - uživatelé vyvíjeného SW - prvek okolí, který komunikuje se systémem.</p> <p>Cílem je skrze <i>aktéry</i> nalézt všechny <i>případy užití</i> systému. Pro samotný systém jsou <i>aktéři</i> bezpředmětní, protože jsou mimo něj.</p>
2.	 <p>Případ užití</p> <p>Případ užití (Use case)</p>	<p>Potomek: <i>Klasifikátor</i></p> <p>Prvek užití vyvíjeného SW, resp. funkce některé jeho <i>třídy</i>. Definuje její chování bez podrobné specifikace vnitřní struktury. Specifikuje řadu akcí a interakce s okolím. [1]</p>
3.	 <p>Svazek (Package)</p>	<p>Potomek: <i>NameSpace</i></p> <p>Obecně se používají ve složitějších diagramech. Zapouzdřuje skupinou <i>prvků</i> modelu (vč. dalších <i>svazků</i>) a slouží k jeho rozumnému rozdělení.</p> <p>Může vlastnit <i>prvky</i>. Vlastnictví je unikátní (žádný jiný <i>svazek</i> nemůže vlastnit daný <i>prvek</i>). Skrz metatřidu <i>ElementImport</i> může obsahovat importované <i>prvky</i>, vlastněné jiným <i>svazkem</i>. <i>Prvky</i> mohou být mezi <i>svazky</i> dostupné i skrz relaci <i>dependency</i>.</p> <p>Zde s objektem <i>PORT</i>: bod interakce mezi <i>klasifikátorem</i> a jeho prostředím nebo (chováním) <i>klasifikátoru</i> a jeho vnitřní části.</p>

Je nutné zvládnout strukturování sady *případů užití* správným nastavením vztahů mezi nimi a nastavení různých priorit jednotlivých *případů užití*.

V praxi se často používá – z hlediska UML nepřesný – ale přehledný hierarchický vztah mezi *případy užití* (rozklad *případů užití* jako by byly *svazky případů užití*). Teoreticky by interakce hierarchie měla být nahrazena mechanismem hierarchie *svazků případů užití*, které zahrnují další *svazky případů užití* a přímo *případy užití*. [4].

K zachování jednoduchosti diagramů se dále oddělují hlavní a alternativní *toky* událostí (zachytí se hlavní tok pro každý *případ užití* a modifikuje se přidáním alternativních *toků*).

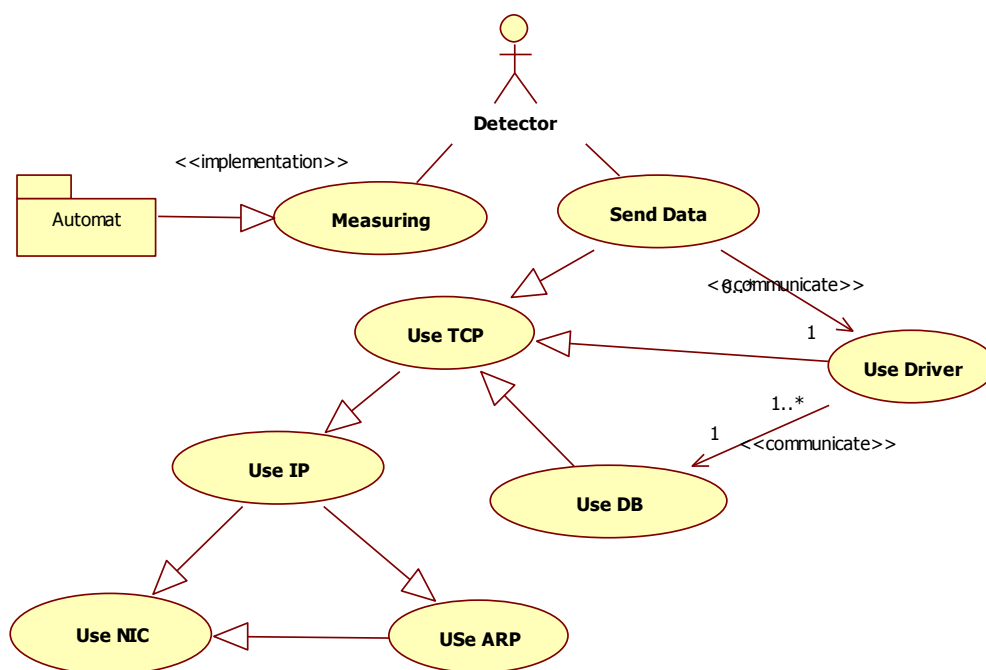
Případem užití při modelování *protokolů* může být řízení komunikace koncových entit z hlediska užití jednotlivých komunikačních vrstev a jejich *protokolů* vrstevnatého modelu – viz kap. 3.

Oba klíčové prvky – *aktér* a *případ užití* jsou potomkem metatřídy *klasifikátor*. Tato vlastnost umožňuje dědičnost – definovat obecný *aktér* a *případ užití* a následně provést jejich specializaci pomocí relace *generalizace*. Z hlediska interpretace *případu užití* představují předchůdce a následovník, propojení relací *generalizace* jeden celek. Dědičný může být i samotný *případ užití*, jako *instance případu užití* (výskyt *případu užití*, který vykonává konkrétní sekvenci *Akcí*) [2].

Prakticky mohou *aktér* a *případ užití* představovat: {datový typ, třídu, interface, uzel, komponentu, subsystém}.

Prvky diagramu užití propojují *relace*, které exaktně definují body spojení (částí) systému a uživatelů a komunikaci mezi nimi. Používají relace: {*Asociace*, *Přímá asociace*, *Generalizace*, *Závislost*, *Začlenění*, *Rozšíření*} – viz PŘÍLOHA 1.

Zdroj: [1], kap. 2, pokud v textu není konkretizováno.



Obr. 5: diagram případu užití pro detektor zasílající měřená data sítě

Obr. 5 ukazuje *diagram příkladu užití* komunikace obecného detektoru s databází. Detektor plní (užívá) dvě primárně funkce – měření veličin a odesílání dat o měření. Funkce měření implementuje KA jako modul a to je modelováno jako *svazek* KA relací *generalizace*. Detektor komunikuje s ovladačem (driver), resp. jeho prostřednictvím s databází, kam změřená data ukládá. Všechny komunikující prvky požívají *zásobník* IP protokolu a síť standardu Ethernet, což modelují *relace generalizace* mezi jednotlivými *prvky případů užití*.

2.6.2 DIAGRAM SPOLUPRÁCE (COLLABORATION DIAGRAM)

Diagram spolupráce je jeden ze dvou typů diagramů **interakce**. Druhým typem je *sekvenční diagram*. Používají se ve fázích analýzy a návrhu *protokolů* – budují „scénosled“ jejich budoucího chování.

Po definici požadovaného chování *případy užití*, pokračuje analýza modelu definicí **kooperace množiny tříd**. Modeluje se **kooperace statické a dynamické struktury množiny objektů** pro každý *případ užití* modelu požadavků (které představují spolupráci *klasifikátorů* – **architekturu systému**). Každý *prvek případ užití* je nahrazen korespondujícím prvkem *objektu, třídy* použité v SW produktu. *Objekty* spojují *relace link*, které reprezentují jejich komunikaci.

Relace definují typy *zpráv*, podmínky a směr jejich zasílání. V této souvislosti:

- **Relace** je spojení odesílatel – příjemce (zahrnuje obousměrný tok).
- **Dialog** je peer-to-peer spojení mezi dvěma koncovými body komunikace, které přetrvává delší dobu.
- **Transakce**: je výměna mezi klientem a serverem; zahrnuje *zprávy* na první žádost odeslané klientem serveru, až po konečné odpovědi odeslané serverem klientu.

Kolekci hlavních Prvků Diagramů spolupráce představují:

- *Prvek objekt (Object)* – viz popis v kapitole 2.6.3.
- *Relace spojení (Link)* – viz PŘÍLOHA 1
- *Relace stimul (Stimul)* – viz PŘÍLOHA 2

Nejčastěji se modelují sekvenční toky (*zpráv, událostí, příkazů*) – ukazují obecnou spolupráci mezi *objekty*. Sekvence se v *diagramu* značí sekvenčním číslem. *Diagramy spolupráce* umožňují modelovat složitější toky, např. opakování a větvení. Pro opakování je doplněn před sekvenční číslo zprávy prefix v podobě opakovacího výrazu (*např.:*

$j:=1..m$). Pro větvení je doplněn prefix s podmínkou větvení (např.: $i>10$). Soubor podmínek se musí vztahovat na všechny kombinační možnosti větvení, resp. opakování.

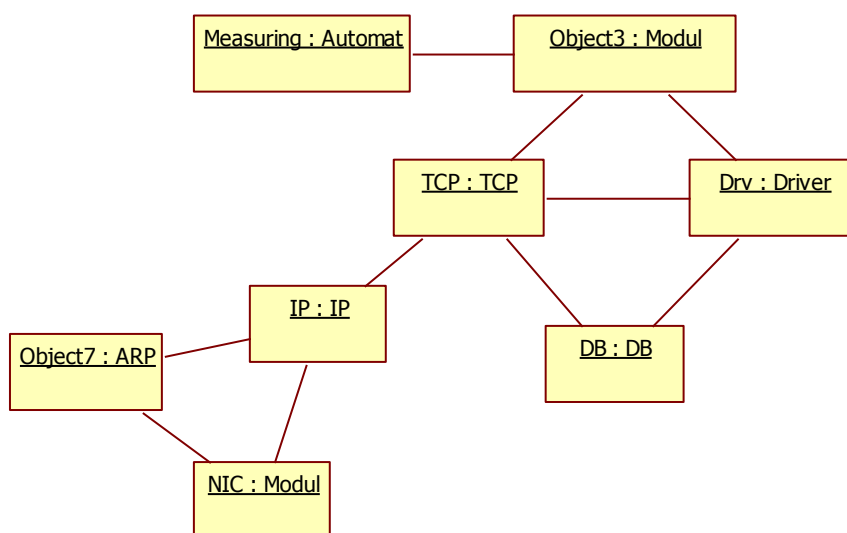
Dále lze modelovat řízení komunikace koncových entit z hlediska užití komunikačních vrstev a jejich protokolů vrstevnatého komunikačního modelu (OSI – viz kapitola 3).

Diagram zobrazuje buď jen modelované části systému, nebo i části systémů, s nimiž model komunikuje v rámci plnění projektované funkce. Pak jde o model end-to-end spolupráce.

V podstatě jde o SW architekturu – soubor partikulárních vazeb kooperace *objektů* modelovaných na základě konkrétního *případu užití*. Studium kooperace jednotlivých *objektů* lze kontrolovat proveditelnost modelu; při odhalení chyb se opakovaně kontrolují a zpřesňují *diagramy užití* i *diagramy spolupráce*, až do odstranění všech chyb.

Reálný *diagram spolupráce* pro netriviální systém bude složitý. Modeluje se při zachování max. jednoduchosti a dělení modelu do více částí. Důležitým hlediskem zjednodušení je definice aplikačního programovacího rozhraní (API) a virtuální spolupráce řízená peer-to-peer protokoly. Obě techniky zapouzdřují řadu jinak detailních funkcí, toků *událostí* a povelů, které takto lze vyjádřit pouze jako *akci* konkrétní funkce konkrétní *třídy*, a to napříč řadou modulů a komunikačních protokolů na různých vrstvách.

Zdroj: [1], kap. 2, pokud v textu není konkretizováno.



Obr. 6: Diagram spolupráce pro detektor zasílající měřená data sítí

2.6.3 DIAGRAM TŘÍD (CLASS DIAGRAM)

Modeluje **statickou strukturu** systému. Klíčové abstrakce tohoto diagramu jsou vztahy mezi *třídami* ve fázi návrhu:

- **Slovník systému:** množina abstrakcí.
- **Spolupráce:** množina *tříd*, *rozhraní* a dalších *prvků*, spolupracující na složitějších funkcích.
- **Schéma:** při modelování *protokolů* se používá pro *perzistentní objekty* (existují nezávisle na *procesu* nebo *stavu*, který je vytvořil), a které udržují konfigurační a podobné informace.

Tabulka 3: Základní prvky diagramů tříd

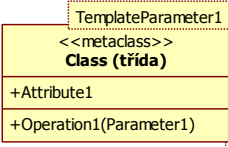
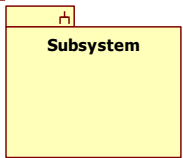

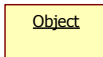
#	Grafický symbol	Popis
1.	 <p>Class (Třída)</p>	<p>Reprezentace množiny <i>objektů</i>, které sdílejí stejné <i>atributy</i>, <i>operace</i>, <i>metody</i>, <i>relace</i> a sémantiku. Může mít množinu <i>rozhraní</i> pro specifikaci všech <i>operací</i>, které mohou <i>objekty třídy</i> poskytovat.</p> <p>Udržuje kolekce <i>TemplateParameters</i> (<i>prvky</i>, vázané v šabloně <i>třídy</i>, které mohou být nahrazeny) a <i>attributes</i> a <i>operations</i>.</p> <p>Zde s nastaveným <i>stereotypem metaclass</i> - viz níže.</p>
2.	 <p>Subsystem</p>	<p>Potomek: <i>Svazek</i></p> <p>Skupina <i>prvků</i>, <i>modul</i> fyzického systému (<i>unit</i>) specifického chování. Má kolekce <i>rozhraní</i> a <i>operací</i> díky dědění ze strany <i>klasifikátoru</i>. Má prvky specifikace a realizace. Prvky specifikace jsou implementovány prvky realizace.</p> <p><i>Svazek</i> je obecnější, nemá implementační povahu; slouží k libovolnému rozčlenění <i>prvků</i> modelu v libovolné úrovni abstrakce.</p>
3.	 <p>Interface</p> <p>Interface (Rozhraní)</p>	<p>Pojmenovaná množina <i>operací</i>, charakterizuje chování nějakého <i>prvku</i>. Obsahuje <i>operace</i>.</p> <p>Nesmí mít <i>atributy</i> a <i>metody</i>, nesmí vstupovat do <i>Asociací</i>. Může být <i>zobecnitelným prvkem</i> a vstupovat do vztahu <i>generalizace</i>.</p>
4.	Svazek (Package)	Viz Tabulka 2, řádek 3
5.	 <p>Objekt</p>	<p>Potomek: <i>třída</i></p> <p>Může mít přiřazeno několik <i>tříd</i> - daný <i>objekt</i> může měnit své vlastnosti v průběhu života na základě přiřazení k dané <i>třídě</i>.</p> <p>Nese <i>klasifikátor</i>, <i>stereotyp</i> a kolekci <i>Attributes</i> s. <i>Slot</i> (šterbina v instanci (slot in instance)) drží konkrétní hodnotu <i>atributu</i> - je reprezentován <i>spojením atributu</i> (<i>AttributeLink</i>).</p>

Diagram tříd využívá k propojování prvků *relace*: {*asociace*, *agregace*, *kompozice*, *generalizace*, *závislost*, *realizace*} viz PŘÍLOHA 1

Další důležitou skupinou jsou prvky **redukce dvojznačnosti**. V podstatě jde o prvek *třída* s určeným *stereotypem*. Následující popis je více konkrétní a vztahuje se k příkladu KA (konečný automat) uvedenému níže v této kapitole:

- **Metatřída (metaclass):** „třída tříd“; nejednoznačnost v modelu řeší použití *metatřídy* místo *třídy*. Pak je jednoznačné, zda jsou *objekty* reprezentované *třídou*, např. zdrojový a cílový stav KA stejné nebo odlišné *třídy*.
- **Parametrizovaná třída (parameterized class):** speciální *třída*, má jeden nebo více nevázaných (unbound) formálních parametrů a *relaci* vztahující se k nim. Sdružuje vázané prvky do šablony formálních parametrů. Vázaný prvek přidá výsledek vazby mezi parametry šablony.
- **Signál (signal):** (vždy) asynchronní podnět, impuls, který komunikuje mezi *instancemi*. Přijímací *instance* zpracovává *signál* dle zavedeného KA. Má *atributy*, které reprezentují parametry *signálu*.
- **Příjem (reception):** specifikuje, jak je daný *klasifikátor* připraven na příjem *signálu*. *Násobnost příjem > signál* je implicitně {1..n}.
- **Výjimka (exception):** potomek *signálu*, je vyvolán prvkem chování zejména v případě chyby systému.
- **Datový typ (data type):** hodnota bez objektové identity. Pokud je *datový typ* výčet, tabulka výčtových *literátů* drží informace jmen a odpovídajících hodnot.
- **Nástroj (utility):** udržuje kolekci *operací* a *atributů*

Zvládnutelná velikost *diagramů* se docílí modelováním menšího rozsahu spolupráce, který popisuje jen některé aspekty systému. Používají se *svazky* a *subsystémy*, které umožňují hierarchicky uspořádat *diagram*, řídí složitost modelu a vyjadřují širší kontext interakce modelu s okolím. Nebo se modelují instance *tříd* k redukci dvojznačnosti (viz výše), pokud je třeba explicitně ukázat dynamický typ *instance* a jiné skryté vlastnosti.

Na Obr. 7 je příklad zjednodušeného *diagramu tříd* KA. Kromě ukázky modelování *diagramů tříd* je tento příklad zařazen k popisu základní statické struktury KA, který jako nástroj představuje modelový základ konstrukce *protokolů*.

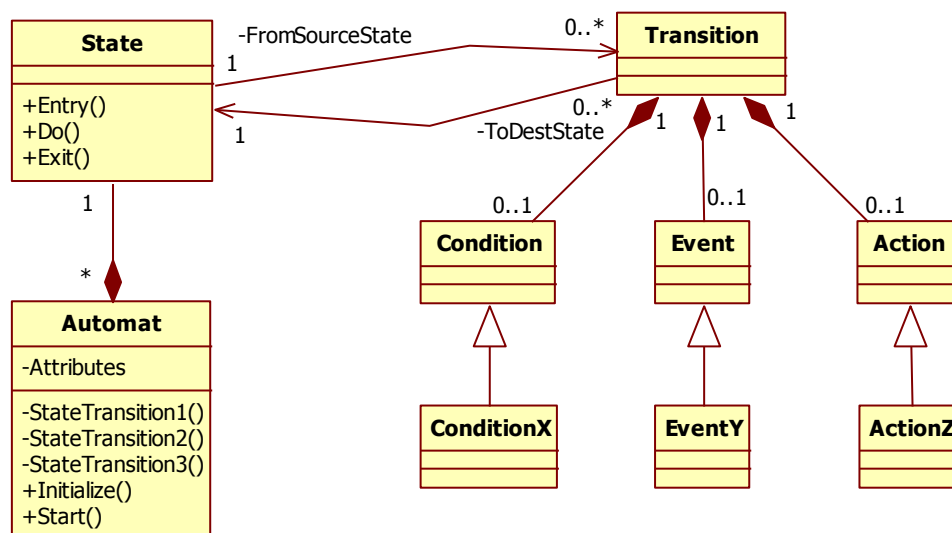
KA sestává z množství *stavů*, to *relace agregace* mezi třídou *automat* a *stav*; *násobnost* z třídy *automat* je (1) a z třídy *stav* je {*}. KA tedy musí obsahovat alespoň jeden technicky relevantní stav.

Dvě *relace asociace*, mezi třídou *stav* a *přechod*, modelují *přechod* zdrojových a cílových *stavů* KA. *Násobnost* z třídy *stav* je (1), protože každý *přechod* stavu musí mít přesně jeden zdroj a jeden cílový *stav*; z třídy *přechod* je *násobnost* {0 .. *}, protože *stav* může mít {0..n} odchozích a příchozích *přechodů*. Klíčové abstrakce ohledně *přechodu stavu* KA modelují tři *třídy*, které představují **obecný KA**:

- *Podmínka*: kontroluje *přechod stavu*
- *Událost*: iniciuje *přechod stavu* a
- *Akce*: *přechodem* tohoto *stavu* vyvolaná

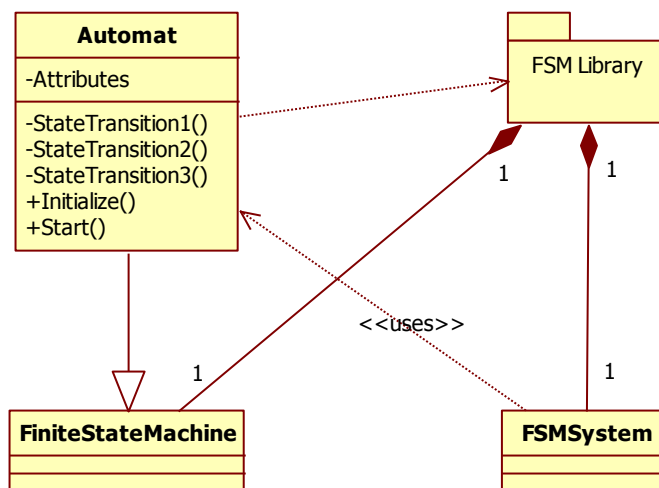
Konkrétní KA definují třídy *PodmínkaX*, *UdálostY*, *AkceZ*, tedy konkrétní *podmínky*, *události* a *akce*, konkrétního KA. Každý *přechod stavu* KA charakterizují tyto prvky, což modeluje *relace agregace* mezi třídou *přechod* a třídami *Stav*, *Událost*, *Akce*. Volitelnost těchto prvků modeluje nastavení *násobnosti relací* na {0..1} ze strany těchto *tříd*.

Kromě akcí, vyvolaných v průběhu *přechodu stavů*, jsou definovány *stavově vázané akce*. Ty se provádí při *přechodu KA z/do* určitého *stavu*, nebo když se KA nachází v určitém *stavu* – viz kap. 2.5.1.



Obr. 7: Diagram tříd konečného automatu

Z hlediska modelování *protokolů* se využívají doménově specifické *diagramy tříd*. Představují obecné konstrukční mechanismy – připravenou architekturu a infrastrukturu tak, že není nutné modelovat KA vždy od začátku. Příkladem je knihovna FSM (finite state machine), jejíž základní diagram *tříd* je na Obr. 8



Obr. 8: Diagram tříd knihovny FSM (KA) pro vývoj komunikačních protokolů

Knihovna je modelována jako *svazek FSMLibrary* a její dvě nejdůležitější *třídy* *FiniteStateMachine* a *FSMSystem*. To, že knihovna obsahuje tyto *třídy*, modelují *relace* mezi nimi a svazkem *FSMLibrary*. *Násobnost* je (1) na obou stranách *relace* (knihovna obsahuje jednu takovou *třídu*).

Protokol modeluje *třída Automat*. Specifičnost typu KA modeluje *relace generalizace* mezi *třídou Automat* a *FiniteStateMachine*. Využitím této *třídy* v podstatě modelování *protokolu* obnáší definovat jeho *stavy* a *přechody stavů*. Jakmile je model dokončen, jeho SW implementace obnáší psát odpovídající rutiny změn *stavů*.

třída Automat závisí na *svazku FSMLibrary*. To modeluje *relace závislosti* mezi *třídou* a *svazkem*. *Automat* je specializací *třídy FiniteStateMachine* a používá ji *třída FSMSystem*.

Druhá důležitá *třída* knihovny je *FSMSystem*. Poskytuje runtime systém pro všechny *protokoly*. Po startu systému hlavní program, zaregistruje daný *protokol* (volá metodu Add s parametrem *Automat*). Jakmile se zaregistruje, *protokol* přijímá, zpracovává a generuje *události (zprávy)*, prostřednictvím schránek, které *třída FSMSystem* poskytuje.

FSMSystem řídí všechny *události*, analyzuje jejich zdroj a cíl a lokalizuje cílový *protokol*. Identifikuje jeho aktuální stav; na základě kódu *události* (typ) určí rutinu *stavového přechodu* a zavolá ji; mechanismus modeluje *relace užití* mezi FSMSystem a Automat.

Zdroj: [1], kap. 3, pokud v textu není konkretizováno.

2.6.4 OBJEKTOVÝ DIAGRAM (OBJECT DIAGRAM)

Objektový diagram představuje speciální typ diagramu. Některé zdroje literatury jej neuvádí jako samostatnou instanci, ale chápou jej jako součást *diagramů tříd, komponent, a implementace*. V případě, že tyto *diagramy* popisují čistě *objekty* a jejich vazby, představují *objektové diagramy* [1], [4]. Nicméně z hlediska modelování *protokolů* jej explicitně popisuje [1] jako prostředek **statického zachycení dynamických dějů**.

Objektový diagram představuje jeden snímek *diagramu spolupráce* v čase – pomyslným zastavením vývoje situace v *diagramu spolupráce*. Lze tak analyzovat určité okamžiky v životě *protokolu* a *objektovými diagramy* vytvářet jejich kombinace a pohledy. Série *objektových diagramů* zachycuje změny hodnoty určitých *atributů*, např. *stavu*. To jsou v modelování *protokolů* informace s nejvyšší hodnotou.

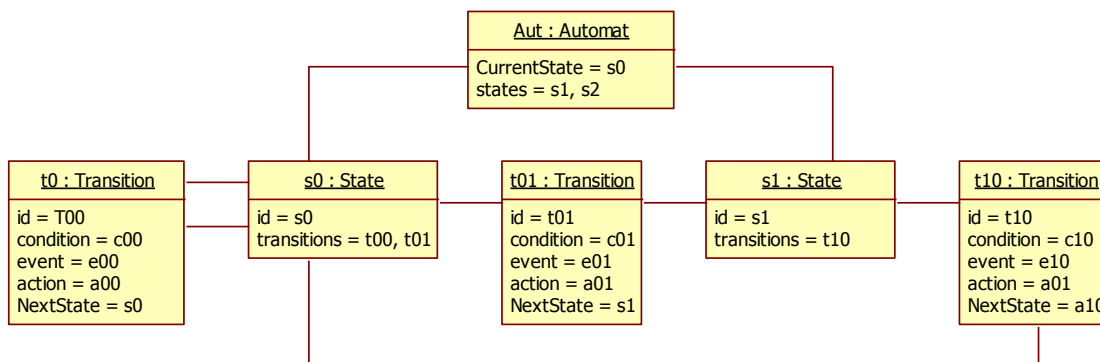
Čistý *objektový diagram* zobrazuje (výhradně) množinu objektů (*instance klasifikátorů*) a jejich vazby. Lze modelovat *klasifikátory*, především k objasnění vztahů *tříd* a *objektů*, dále *svazky* a *podsystemy* k řízení složitosti.

I zde platí omezit model jen na určitý rozsah problematiky systému a její detaily. Grafické prvky jsou stejné jako pro *diagramy tříd*. V praxi se používají nejčastěji pouze dva – *objekt* (viz Tabulka 3, řádek 5) a *reference link* viz PŘÍLOHA 1

Objektový diagram snižuje dvojznačnost statické struktury, tak že:

- 1) Modelování *instancí klasifikátorů* přináší hlubší poznání vztahů mezi nimi. (Pokud se např. modelují pouze *třídy*, nemusí být jasné, jak přesně probíhá signalizace)
- 2) Ukazuje klíčové hodnoty *atributů tříd*. Tak lze přesně poznat realitu. (Např. identifikace určitých *stavů* jednotlivých *protokolů*, zpřesňuje chápání jejich očekávání od okolí, se kterým interagují (jiné *protokoly*, obslužná zařízení atd.)).
- 3) Poskytuje jasný pohled na vrstvy a *protokoly* OSI modelu, jak je vyžadují a používají *hostitelské systémy* a *síťové služby* – viz kap. 3. Graficky, je komunikační síť, uspořádána přehledněji než na jiných typech diagramů.

Obr. 9 je příklad *objektového diagramu*, který rozvíjí KA z předchozí kapitoly. Objekt KA, pojmenovaný Aut, je *instancí třídy Automat*. Objekty S0 a S1 představují *stavové přechody* a jsou *instancemi třídy stav*. Výchozí stav KA je S0.



Obr. 9: Objektový diagram KA

Stavový objekt *s0* má kolekci dvou objektů přechodu stavu – *T00* a *T01*, které jsou *instancemi třídy přechod*. Analogicky objekt *s1* má jeden přechod stavu – *T10*. objekty přechodu stavů $\{T00, T01, T10\}$ modelují přechody KA takto: $\{S0>S0, S0>S1, S1>S0\}$.

atributy objektů přechodu stavu $\{id, stav, událost, činnost, nextState\}$ identifikují ochrannou podmínku, která hlídá událost, která spouští přechod a akci, kterou přechod vyvolá; dále identifikaci následujícího přechodu. Stav, Událost, Akce jsou instance třídy Stav, Událost, Akce. Hodnoty těchto atributů jsou instance tříd, které se specializují od Třídy Stav, Událost, Akce. (Např. hodnoty atributu stav $\{con00, con01, con10\}$ jsou instance tříd, $\{condition00, condition01, condition10\}$, které jsou ve skutečnosti potomky třídy condition).

Tento styl modelování umožňuje používat **polymorfismus** – nejsilnější abstrakční přístup objektově orientovaného modelování a programování. Znamená schopnost zaujímat více forem v různých významech. Např. jeden objekt může mít několik metod téhož jména, které však provádějí různou činnost a odlišují se počtem nebo typem parametrů.

Zdroj: [1], kap. 3, pokud v textu není konkretizováno.

2.6.5 SEKVENČNÍ DIAGRAM (SEQUENCE DIAGRAM)

Sekvenční diagramy představují druhý typ interakčních diagramů. Jsou sémanticky ekvivalentní diagramům spolupráce, což přináší mapování 1:1 mezi oběma typy.

Sekvenční diagramy akcentují časovou posloupnost zpráv. *Diagramy spolupráce* akcentují strukturální organizaci množiny objektů.

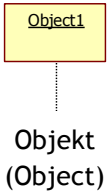
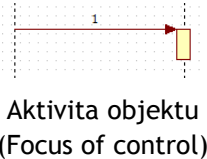
Interakční diagramy **popisují** množinu objektů a jejich vztahy **zprávami**, které si objekty vyměňují. *Sekvenční diagramy* vizualizují dynamické chování v kontextu scénářů *případů užití*. Budují „scénosled“ budoucího chování modelovaného *protokolu*. Obecně jsou vhodnější k modelování sekvencí *událostí*, jednoduchých iterací a větvení; alternativně k modelování složitých iterací a větvení, a pro vizualizaci více souběžných *toků*.

Diagram připomíná tabulku; sloupce (osa y) se vztahují k jednotlivým *objektům* a řádky (osa x) představují tok zpráv mezi *objekty*, které se podílejí na interakci. *Objekty* jsou řazeny zleva počínaje těmi, co iniciují interakci. *Zprávy*, které se zasílají mezi *objekty*, jsou řazeny v čase podél osy y (řadí se nad sebe). *Sekvenční diagramy* mají dvě klíčové vlastnosti, které je odlišují od ostatních typů diagramů:

Modelování **mutací** *Objektů* v aktuálním *Stavu*, roli a hodnotách *Atributů* má tyto postupy:

- 1) umístění nové kopie *Objektu* do *Diagramu* a akcentací změny *Stavu* propojením stávající a nové kopie *Objektu* *Přechodem stavu* (Stereotyp <<become>>). Postup lze opakovat pro sérii změn.
- 2) umístění nové kopie *Objektu* na *Čáře života* původního *Objektu* a přímo ukázat změnu *Stavu*, *Role* nebo hodnoty *Atributu*.

Tabulka 4: Základní prvky sekvenčních diagramů

#	Grafický symbol	Popis
1.		Viz Tabulka 3, bod 5, vykreslovaný vč. čáry života (object lifeline): přerušovaná svislá čára; představuje existenci <i>objektu</i> po určitou dobu. Začíná přijetím <i>zprávy</i> <i>stereotypu</i> <<create>> a končí přijetím <i>zprávy</i> <i>stereotypu</i> <<destroy>>. Konec života <i>objektu</i> je indikován značkou "X" na čáře života; většina objektů ale existuje po celou dobu interakce.
2.		Zdvojení úseku čáry života; doba, po kterou je <i>objekt</i> aktivní (pracuje). Stínováním části odpovídajícího regionu se explicitně modeluje aktivita <i>Objektu</i> (kde skutečně probíhá výpočet).

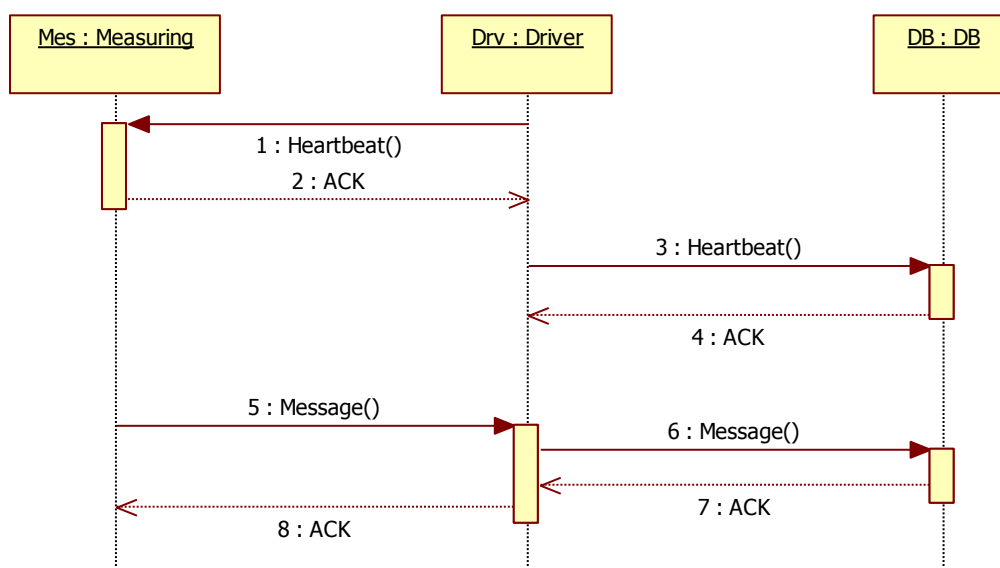
Stimul: představuje komunikovanou *zprávu*; kromě jiných nese vlastnost *ActionKind* (*Typ akce*) a dle zastávané role nabývá hodnot {*create*, *destroy*, *call*, *send*, *return*}. V souladu

s nimi mění i grafickou podobu. V *sekvenčních diagramech* představuje *relaci* mezi *objekty*. Viz PŘÍLOHA 1.

Šířka a výška *diagramů* je omezena potřebami dodržení přehlednosti a dalšího zpracování výsledného dokumentu. Proto je i zde důležité řízení složitosti modelu a jeho techniky:

- 1) Tok *událostí* a *zpráv* se přeruší, aby pokračoval v jiném *diagramu*. Přerušení se volí logicky, např. na začátku nebo na konci určité fáze komunikace.
- 2) Oddělené modelování určených *tříd zpráv* a *událostí*; např. řídicí zprávy a ošetření chyb a výjimek se oddělí do vlastních *tříd*, ty se modelují v jiných *diagramech* a do daného *diagramu* se umístí jako *svazky*.
- 3) Modelují se individuálních virtuálních interakce omezené skupiny *objektů*, které pokrývají určitou část komunikace – analogická technika k bodu 1.

Zdroj: [1], kap. 3, pokud v textu není konkretizováno.



Obr. 10: Sekvenční diagram komunikace detektoru

Obr. 10 vychází z příkladu z kap. 2.6.8. Zachycuje základní komunikaci mezi detektorem (objekt Mes třídy Measuring) a řídicím systémem (komponenty Driver a DB). Driver pracuje mj. jako řadič, který kontroluje dostupnost všech *komponent* pravidelným zasíláním kontrolní zprávy heartbeat a kontrolou korektního doručení odpovědi (ACK; Acknowledgement). Pokud detektor získá data, odešle je přímo (asynchronní režim) a očekává doručení odpovědi. Pokud ji nedostane, spustí obsluhu chyb (viz diagram aktivit v kap. 2.6.6).

2.6.6 DIAGRAM AKTIVIT(ACTIVITY DIAGRAM)

Interakční diagramy nejsou konečné artefakty fáze návrhu *protokolu*. Vzhledem k problémům níže se návrh založený jen na interakčních diagramech považuje za neúplný:

- Jsou zpravidla neúplné, bez ohledu na pečlivost projektanta. Kromě lidského faktoru způsobuje chyby reálná potřeba dělit *diagramy* do logických celků, vyjádřených na oddělených výkresech, na kterých vynechané části interagují jako *svazky* – viz kapitoly 2.6.2 a 2.6.5. Kromě toho jejich primární úlohou je modelovat nejdůležitější aspekty a scénáře návrhu (k jejich efektivní využití v dalších krocích), při zachování udržitelné míry složitosti a srozumitelnosti.
- Popisují jen vnější chování *objektů* modelu, vnitřní chování zůstává neznámé.
- Konstrukce *protokolů* je zvláště citlivá na specifikaci časovačů a komplexnost nepředvídaných chyb *svazků* interakčních diagramů
- *Svazky* kompletních interakčních diagramů se stávají objemné, a proto těžko pochopitelné (i program pro jednoduché aritmetické výpočty může produkovat obrovské množství exekučních případů)

Diagram aktivit představuje pracovní postup (work flow), resp. vývojový diagram. Obrací pozornost na **vnitřní chování** *objektů* interakčních diagramů a specifikuje:

- *Aktivitu*, které musí být provedeny k zajištění požadovaného vnějšího chování
- Model chování každého jednoho *objektu* – *tok řízení* aktivit. Chování jednoho objektu se rozdělí do množiny operací a modeluje se *tok řízení* těchto operací individuálně. Nejnižší úrovní modelování je proto úroveň operací daného objektu.
- Model chování skupiny *objektů* – popisuje *tok řízení* aktivit **mezi** těmito *objekty*

Modely pracovních postupů specifikují:

- **Vnější chování:** *Rozhraní a Protokol* mezi *Objekty* – jako posloupnost *Zpráv* mezi *Objekty* zasílaných. Tvoří jej model dat a tok (stavů) *Objektů*.
- **Vnitřní chování objektů:** sérii stavů *Aktivitu*, kterými *Objekty* procházejí. Tvoří jej model toku řízení přes *Objekty*.



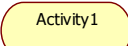


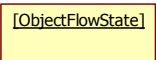
Uvnitř jednoho *objektu* nebo ve skupině *objektů* se modelují:

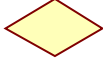
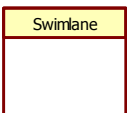
- jednosměrný a obousměrný tok řízení,
- více paralelních *toků*,
- větvený tok řízení s odbočkami

Nejmenší jednotkou aktivity je *Stav akce (action state)*, který vyvolá *akci*, následovanou *přechodem* do jiného *stavu* na základě jejího výsledku. Jde o **nedělitelnou výpočetní operaci** – např.: {vytvoření objektu a zrušení, volání operace objektu, vrácení hodnoty operace, zaslání signálu, příjem signálu, vyhodnocení výrazu, provedení jednoho příkazu}.

Ačkoliv *stav akce* spotřebuje malé množství (procesorového) času, bere se jeho konečné množství v úvahu zejména v modelech real-time systémů (*protokolů*).

Tabulka 5: Základní prvky diagramu aktivit

#	Grafický symbol	Popis
1.	 Počáteční stav (Initial state)	Pseudostav Viz kap. 2.5.2
2.	 Konečný stav (Final State)	Potomek: <i>Stav</i> Pokud diagram neobsahuje tento symbol, modelované aktivity pokračují do nekonečna. Diagram alternativně může obsahovat jeden nebo více <i>konečných stavů</i> . [1] Dále - viz kap. 2.5.1.
3.	 Stav akce (ActionState)	Potomek: <i>State (SimpleState)</i> Mohou probíhat souběžně (concurrently) - určeno <i>atributem isDynamic</i> (TRUE). Má význam v souvislosti s hodnotami ostatních <i>atributů</i> . <i>DynamicMultiplicity</i> je limit <i>násobnosti</i> počtu souběžných <i>aktivit</i> .
4.	 Subaktivita-stav (SubactivityState)	Potomek: <i>SubmachineState</i> Modeluje vnořování (volání) <i>stavových strojů</i> mezi sebou. Reprezentuje vykonání ne-atomické <i>aktivity</i> složené z jiných <i>aktivit</i> (vykoná vnořený <i>diagram aktivit</i> - umožňuje je přirozeně skládat stejně jako <i>stavové stroje</i>).
5.	 Synchronizace (Synchronization)	Pseudostav Prvek rozdělení a spojení <i>přechodů (toků řízení)</i> . Viz kap. 2.5.2. Párový prvek - každá úroveň vnoření <i>řízení toku</i> začíná a končí tímto prvkem. To je jediné omezení počtu úrovní vnoření, kromě udržitelné složitosti a srozumitelnosti modelu. [1]
6.	 Objektový tok-stav (ObjectFlowState)	Potomek: <i>State (SimpleState)</i> <i>Vrchol stavového stroje, (Stav)</i> . Vyjadřuje objektový tok mezi dvěma <i>aktivitami</i> , tj. <i>stav objektu po dokončení jedné aktivity, která předchází uvedení objektu do nového stavu a před zahájením přechodu do stavu následujícího</i> . ObjectFlowState se vloží mezi oba <i>přechody</i> ; vidí jeden

#	Grafický symbol	Popis
		ClassifierInState, jehož Stav reprezentuje. Může být výsledkem <i>stavů aktivity</i> a může být používán jinými <i>stavy aktivity</i> . Obecně, každá změna <i>atributu</i> (parametru) <i>objektu</i> znamená změnu jeho <i>stavu</i> a tím vznik jeho nové <i>stavové instance</i> , reprezentované tímto <i>prvkem</i> . [1] Alternativně lze bez použití ObjectFlowState modelovat přímý vztah dvou <i>aktivit</i> jako <i>přechod</i> mezi dvěma <i>stavy</i> . [1]
7.	 Rozhodnutí (Decision)	Pseudostav Uzel, který přijímá {1..N} unikátních příchozích objektových toků nebo <i>toků řízení</i> a vybere {1} odchozí z {1..N}. O výběru odchozího toku rozhoduje <i>Ochranná podmínka</i> . Někdy může mít definován rozhodovací vstup, který redukuje nadbytečné výpočty <i>ochranných podmínek</i> tak, že se nejprve hodnotí všechny příchozí toky. [5]
8.	 Plavební dráha (Swimlane)	Používá se k oddělení jednotlivých stran v pracovních postupech (work flow). Může zapouzdřovat (reprezentovat) <i>třidu</i> nebo skupinu <i>tříd</i> . [1]

Přechody mezi jednotlivými *stavy aktivity* mohou být bez *spouštěče* (*triggerless*); tzn., že nejsou spuštěny jinými *akcemi*. Také nemusí mít *ochrannou podmínku*, pokud jejich zdroji nejsou (bloky) *rozhodnutí*.

Obecně platí, že *stavy aktivity* jsou operace (procedury, funkce), které zahrnují spustitelné výrazy nebo volání jiných operací, včetně rekurze.

Větvený tok *přechodů* zahrnuje více vstupních a výstupních *ochranných podmínek*, které tok větví. *Podmínky* představují *spouštěče* vstupních a/nebo výstupních *přechodů stavů*, jsou vázány na danou *aktivitu stavu*, kde jsou vyhodnocovány (spouštěny) na základě odpovídajících událostí. *Ochranné podmínky* mají tyto vlastnosti:

- Jsou v *přechodech*, kde se tok řízení větví povinné [4]
- Volí se disjunktne (vzájemně se vylučují) [4]. Logika *podmínek* se tak nepřekrývá a tok řízení je jednoznačný.
- Musí krýt všechny možnosti (variace); to zaručuje ochranu *toku řízení* proti uváznutí (deadlock)
- Dojde-li k současnému splnění všech *podmínek*, je další *přechod* libovolný [4].

- Pokud ani jedna *podmínka* není pravda, je **chyba v** modelu [4].

Analogicky se modulují smyčky (cykly v kontextu vyšších programovacích jazyků).

Uvedené premisy vyžadují (pokud možno) absolutní komplexnost modelů a specifikací *aktivit*, které popisují chování systému. Bezpečný způsob, jak je prakticky pokrýt je použít rozhodování omezené na dva odchozí *přechody* a jeden z nich chránit (spouštět) klíčovým slovem *ELSE*. Zvláštní pozornost vyžaduje rozhodování s více odchozími přechody, které nechrání explicitní výraz *ELSE* a použití negativní logiky, kdy výsledek operace *FALSE* **nevyvolá** jen její návratová hodnota, ale – obecně – nečinnost procesu nebo jiný faktor. To způsobuje **dvojnáčnost**, kterou *diagramy aktivity* mají odstraňovat. Např. jde o stav, kdy hodnota *FALSE* jako výsledek nedoručení zaslané datové jednotky (DU) vznikne nejen při obdržení *NACK* (Negative Acknowledge), ale např. i poškozením *ACK* a *NACK*.

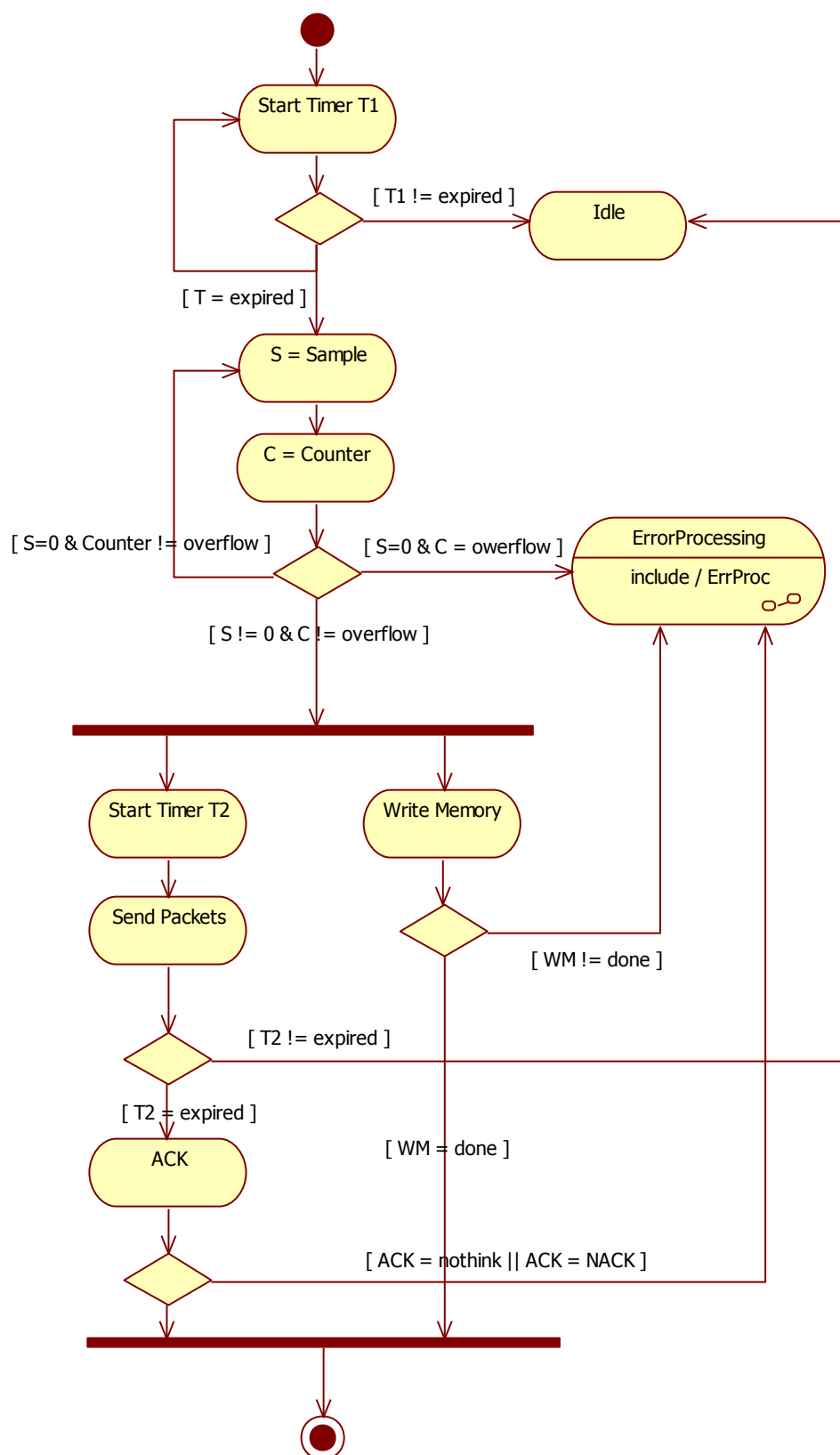
Události spojené s příjmem zpráv se modelují jako ochranné výrazy, zatímco vyvolané akce se modelují jako *akční stavy*.

Další důležitá technika v *diagramech aktivity* je souběžné řízení – **modelování paralelních činností** – využívá se prvek *synchronizace* – viz Tabulka 5, bod 5.

V *protokolech* jde typicky o obousměrnou (full-duplex) komunikaci. *Aktivita* (jako jeden tok) naváže komunikaci s protější stranou; pak se řízení toku rozdělí na dvě paralelní *aktivity* (příjem a vysílání dat). Jakmile jedna z paralelních aktivit skončí, čeká v synchronizačním bodě na dokončení ostatních paralelních aktivit a paralelní řízení toku se spojí do jednoho, který dokončí zbylé činnosti.

Zdroj: [1], kap. 3, pokud v textu není konkretizováno.

Na Obr. 11 je příklad diagramu aktivit (obecného) detektoru, který může být založený na modelu v příkladech popisovaného KA. Časovač T1 v pravidelných intervalech spouští odběr vzorků ze senzoru. V případě výskytu chyby (nulová hodnota) opakuje měření v nastaveném počtu pokusů. Pokud hodnotu nezíská, přejde do STAVU Sub-aktivity ErrPROC, Sub-stoj obsluhy chyb. Pokud vzorek získá, rozvětví zde činnost a zahájí komunikaci skrz připojenou síť, odešle paket a čeká na potvrzení (ACK). Pokud ACK nedorazí, znovu volá Sub-stroj obsluhy chyb. Paralelně uloží naměřený vzorek do vnitřní paměti detektoru. Pokud zápis selže, volá Sub-stroj obsluhy chyb. Z hlediska *zásobníku protokolů* může předávání zpráv, obsluha chyb, konfigurace detektoru a další činnosti probíhat různými *protokoly* a na různých portech.



Obr. 11: Diagram aktivit detektoru

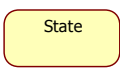
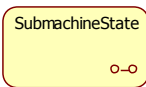


2.6.7 STAVOVÝ DIAGRAM (STATECHART DIAGRAM)



Stavové diagramy modelují životní cyklus objektu (instance třídy), nebo případu užití. Kladou důraz na chování objektů řízené událostmi a jsou vhodné k modelování chování celých stavových objektů, zejména řízených událostmi (zprávami). Byly vyvinuty k modelování KA, jsou tedy vhodné i k modelování protokolů.

Oproti tomu *diagramy aktivity* akcentují tok *událostí* a *stavů aktivity* a jsou vhodnější k modelování jednotlivých operací.

Společný rys obou typů diagramů je, že jsou sémanticky ekvivalentní a zaměřují se na **kompletní model chování**. Poskytují dva různé pohledy na stejné chování a srovnatelnou úroveň detailů. *Stavové diagramy*, pokrývají variace podmínek implicitně, protože používají prvky *rozhodování* zřídka; *diagramy aktivit* (viz kap. 2.6.6), mají toto pokrytí explicitní – prvky *rozhodování* používají široce.

Tabulka 6: Základní prvky stavového diagramu

#	Grafický symbol	Popis
1.	 Stav (State)	Potomek: <i>Stavový vrchol</i> Z hlediska StarUML představuje Jednoduchý a Kompozitní stav. Dále viz kap. 2.5.1
2.	 Stav Sub-stroje (SumachineState)	Potomek: <i>Kompozitní stav</i> <i>Sub-stroj</i> významově odpovídá <i>kompozitnímu stavu</i> . Použití <i>sub-stroje</i> odkazuje na jiný, vnější, stroj, který může zavolat <i>stav sub-stroje</i> jako proceduru obsahující pokračování dalších <i>přechodů</i> uvnitř <i>sub-stroje</i> . Vstupní a výstupní body volání definuje <i>substate</i> - potomek <i>stavu</i> ; propojuje <i>stroj</i> , který volá a <i>stroj</i> , který vykonává.
3.	Počáteční stav	Viz Tabulka 5, řádek 1 a kap. 2.5.2
4.	Konečný stav	Viz Tabulka 5, řádek 2 a kap. 2.5.1.
5.	 Křižovatka (Junction Point)	<i>Vrchol</i> k zřetězení více <i>přechodů</i> . Používá se ke konstrukci složené cesty <i>přechodů</i> mezi <i>stavy</i> . Sbírá více příchozích <i>přechodů</i> do jednoho odchozího, který reprezentuje sdílenou cestu. Mohou být použity k rozdělení jednoho příchozího <i>přechodu</i> do více odchozích segmentů řízených <i>ochrannou podmínkou</i> . Větvě, kde je <i>ochranná podmínka FALSE</i> se neuplatní. [5]
6.	 Volba (Choice Point)	Viz kap. 2.5.2

#	Grafický symbol	Popis
7.	 Mělká historie (Shallow History)	Viz kap. 2.5.2 Ukládá pouze kontext první úrovně vnoření z <i>kompozitního stavu</i> [1]
8.	 Hluboká historie (Deep History)	Viz kap. 2.5.2 ukládá kontext vnitřních stavů v jakékoli úrovni vnoření [1] <i>Hluboká a mělká historie</i> se chovají jako stavová paměť (buffer) <i>kompozitního stavu</i> , V obou případech by měly být označeny implicitní a iniciační stav, kam přejdou <i>přechody</i> , pokud nebyly <i>kompozitní stavy</i> aktivní a tyto vrcholy se nemohly - jako paměť - naplnit.
9.	Synchronizace	Viz Tabulka 5, řádek 5 a kap. 2.5.2

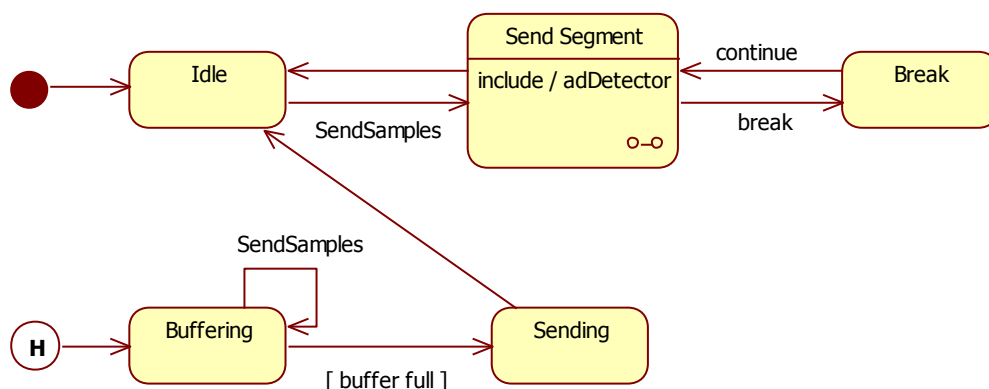
Prvky se propojují *relací transition* – viz PŘÍLOHA 1

Složitost se řídí hierarchickým uspořádáním vnoření *stavů*, které může jít teoreticky do neomezené hloubky. Vznikne *kompozitní stav* (viz kap. 2.5.1), který zahrnuje *jednoduché* a další *kompozitní stavy* a:

- **Sekvenční substavy:** *objekt* může být v určitém okamžiku pouze v jednom z nich.
- **Souběžné substavy:** *objekt* je v určitém okamžiku ve všech souběžných *substavech* aktivních v daném bodě.

Kompozitní stav pracuje **bez ukládání kontextu** – paměti. *Přechody* z externích *stavů* do *sub-stavů* (*kompozitních stavů*) nejsou povoleny; tedy je informace o *toku řízení* ztracena a zpracování je restartováno **od začátku**. Má-li být *kompozitní stav* schopen restartu **od bodu přerušení**, používá se *prvek stav historie*. Ten po aktivaci restartuje *operaci* v bodě přerušení a současně představuje *typ výchozího stavu*, který je cílem pro *přechody* z externích *stavů*. *Mělká historie* ukládá zevní vnořené *stavy* a *hluboká historie* ukládá vnitřní vnořené *stavy*.

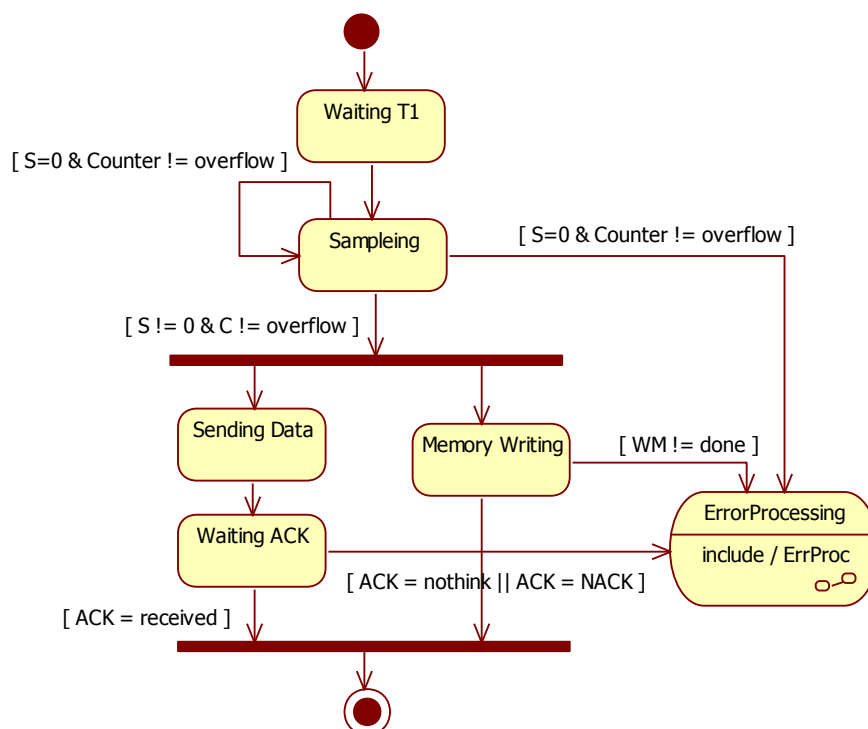
Obr. 12 modeluje příklad využití mělké historie. Odesláním vzorku dat KA přejde do kompozitního stavu Send Segment, který začíná stavem mělké historie (spodní část obrázku), ve kterém jsou uložena předaná data a zajistí jejich odeslání na základě nezávislých podmínek (Zdroj: [1], str. 94).



Obr. 12: Příklad využití paměťového stavu mělké historie v kompozitním stavu

Je podporováno modelování paralelních aktivit, pomocí souběžných sekvencí *sub-stavů* v *kompozitním stavu*. Používá se, je-li chování jednoho z paralelních toků řízení, ovlivněno *stavem* jiného nebo je-li chování paralelních toků řízeno *zprávami*, které si vyměňují. Sekvence (obvykle) začíná *východím stavem* a končí v *konečném stavu*. Přejít z *externího stavu* do *kompozitního stavu* se rozvětví do paralelních *sub-stavů*, které se na konci spojí přechodem z *kompozitního stavu* do *externího*.

Zdroj: [1], kap. 3, pokud v textu není konkretizováno.



Obr. 13: Stavový diagram detektoru

Obr. 13 ukazuje zjednodušený *stavový diagram* detektoru z příkladu v kap. 2.6.6. Z výchozího stavu je spouštěn časovač T1. Po jeho vypršení přejde detektor do stavu vzorkování měřené veličiny, které se opakuje dle nastavené podmínky při chybě snímání. Pokud data nejsou získána, detektor přejde od stavu ošetření chyb, jinak pokračuje do stavu odesílání a ukládání naměřených hodnot. Po odeslání naměřených dat čeká na potvrzení, pokud jej neobdrží, přechází do stavu ošetření chyb. Paralelně ukládá data do paměti, při selhání přechází do stavu ošetření chyb. Při správné činnosti KA, který detektor reprezentuje, přejde do finálního stavu, ze kterého může být znovu iniciován.

2.6.8 DIAGRAM ROZMÍSTĚNÍ ZDROJŮ (DEPLOYMENT DIAGRAM)

Diagramy rozmístění zdrojů (dále DRZ) modelují rozmístění *komponent*, jejich *instancí*, *objektů* a *svazků* na *uzlech* a jejich *instancích*. **Komponenta** je část systému, která implementuje množinu *rozhraní*. Typicky modeluje fyzický *svazek* logických *prvků* (*třídy*, *rozhraní* a *spolupráci* – nejčastěji spustitelné soubory, knihovny, tabulky a dokumenty).

Uzel je fyzický *prvek*, který modeluje výpočetní platformu, zahrnující sadu zdrojů, jako paměťové banky, sběrnice, I/O kanály, radiče, procesory apod. – nejčastěji osobní a sálové počítače, vestavěné radiče, mobilního telefonu, síťové prvky atd.

Pro modelování *protokolů* se DRZ používají zejména:

- K identifikaci *uzlů* a konfigurace sítí
- K identifikaci návrhu *subsystémů* a *rozhraní*

SW architektura úzce souvisí se strukturou fyzické sítě, která může někdy být neměnná a pak určuje rozložení funkcionality na síťových *uzlech*, stejně jako výběr aktivních *tříd*. Pokud je předmětem návrhu *SW architektura* a síťová struktura, pak může konkrétní síťová struktura přinést vhodnější *SW architekturu* a systémové řešení.

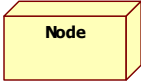
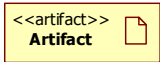

DRZ lze modelovat dvěma přístupy:

- *uzly* sítě se kreslí jako kostky, propojí se *relací asociace* a řeší se rozmístění jednotlivých *komponent* na těchto *uzlech*, které se s *uzly* propojí *relací závislosti*
- instance *uzlů* se popíší jmény *komponent*, které jsou na nich nasazeny

Podobně *subsystémy* a *rozhraní* se kreslí jako *svazky* s odpovídajícím *rozhraním*. Uspořádají se do hierarchických vrstev (např. aplikačně-specifické, aplikačně-obecné, middleware, systémový SW). Nakonec se modeluje, která *rozhraní* (služby) jsou

poskytována, kterými *svazky* nebo *komponentami*, a které *svazky* a *komponenty* používají služby poskytované na těchto *rozhraních*.

Tabulka 7: Základní prvky diagramu rozmístění komponent

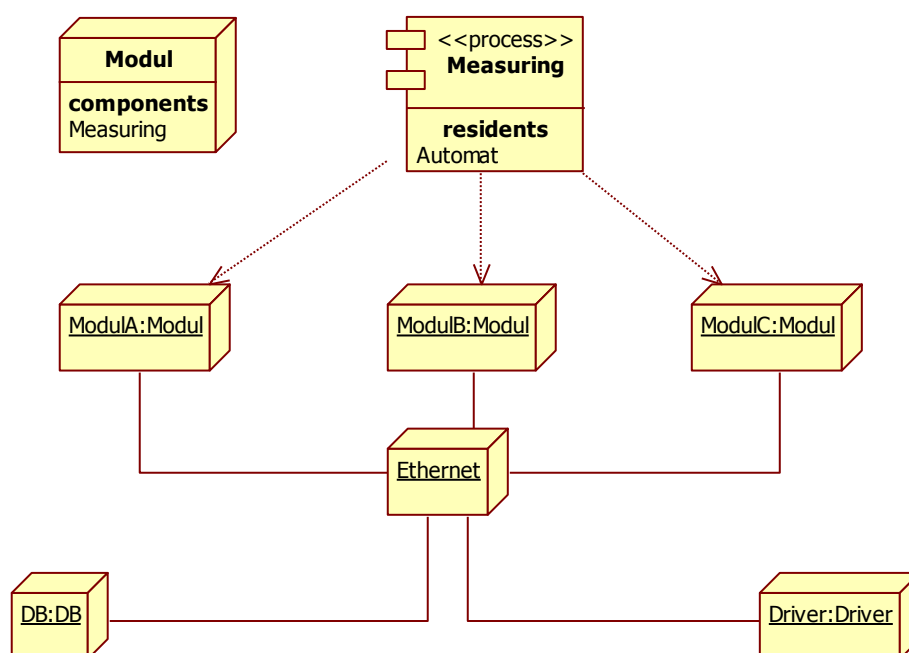
#	Grafický symbol	Popis
1.	Svazek	Viz Tabulka 2, řádek 3
2.	 Uzel (Node) Instance Uzlu (NodeInstance)	<p>Fyzický prvek - má paměť a procesor. Implementuje konkrétní <i>komponenty</i> rozmístěné do <i>uzlu</i> (vztah agregace). Komponenta realizuje (implementuje) různé <i>prvky modelu</i>, např. <i>třídy</i>, <i>rozhraní</i>, <i>atributy</i>, <i>operace</i> atd. (vztah agregace N <i>prvků modelu</i> v <i>komponentě</i>).</p> <p>Na Instanci Uzlu sídlí (residueje) několik Instancí Komponent.</p>
3.	 Artefakt	<p><i>Klasifikátor</i> reprezentující entitu - informaci, <i>operaci</i> používanou nebo produkovanou modelovaným SW. <i>Instance artefaktu</i> je umístěna do <i>instance uzlu</i> a může obsahovat další, vnořené <i>artefakty</i>.</p> <p>Má kolekci <i>operaci</i>, která dále udržuje kolekci <i>parametrů</i>, a přijatých <i>signálů</i>. Nese jméno fyzického souboru, který v modelu představuje. Jde o soubory, skripty, spustitelné soubory, tabulky databází [5].</p>
4.	 Komponenta (Component)	<p>Podtyp <i>Klasifikátoru</i>, fyzická, vyměnitelná část systému, svazuje dohromady určitou implementaci a realizuje množinu <i>rozhraní</i>. Zahrnuje SW (zdrojový, binární, spustitelný, skript příkazové soubory)</p> <p>Jako potomek <i>Klasifikátoru</i> může mít <i>Atributy</i> a <i>Operace</i>.</p>
5.	Svazek	Viz Tabulka 2, řádek 3.
6.	Objekt	Viz popis v kapitole 2.6.3.

Prvky DRZ se propojují *relacemi asociace, závislosti, link, agregace* – viz PŘÍLOHA 2.

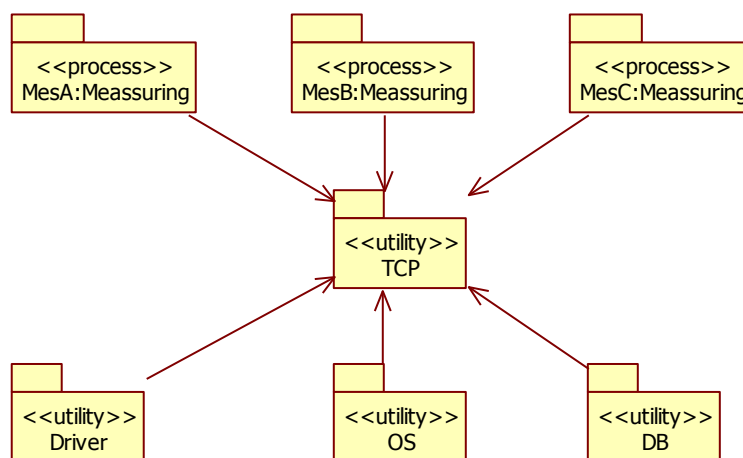
Zdroj: [1], kap. 3, pokud v textu není konkretizováno.

Obr. 14 je příklad HW konfigurace technologického měření. Hostitelský HW představuje obecný „modul“ s vlastnostmi *uzlu*. Síť obsahuje tři *instance uzlu Modul*, které představují fyzické senzory. V každém residueje *instance komponenty Measuring*, která je *instancí třídy Automat* z předchozích příkladů a představuje firmware senzorů. Moduly jsou připojeny k síti Ethernet. *Instance DB* ukládá získaná data a *instance komponenty Driver*, představuje SW ovladač pro příjem a zpracování dat.

Připojení modulů Ethernetu modeluje *relace link*, závislost modulů na firmware Automat, *relace závislosti* od komponenty Measuring.



Obr. 14: DRZ topologie měřicího systému se třemi detektory a řídicím SW



Obr. 15: DRZ SW komponent měřicí soustavy

Obr. 15 ukazuje SW komponenty předchozího příkladu. Komponenty Measuring je realizována instancemi MesA až MesC a představují systémovou vrstvu SW. Driver a DB pracují na aplikační vrstvě a operačního systému (OS) a TCP na systémové.

3 VRSTEVNATÁ ARCHITEKTURA KOMUNIKAČNÍCH MODELŮ

Zdroj: pokud není v textu konkrétně uvedeno jinak, vychází celá kap. 3 z [11], resp. [12].

3.1 Význam vrstevnatosti komunikačního modulu

Technika zjednodušení a optimalizace návrhu rozdělením a specializací *protokolů* (a souvisejících aplikačních procesů) do *funkčních vrstev*, z nichž každá plní určitou oblast (*třidu*) funkcí, pro které je navržena a optimalizována. V rámci každé *vrstvy* existuje jeden nebo více subjektů, které mohou provádět funkčnost. [10]

Každá *vrstva* má přímou interakci pouze s *vrstvou* bezprostředně pod ní a poskytuje pracovní prostředky vrstvě nad ní. *Vrstvy* jsou vzájemně nezávislé. Jsou proto modifikovatelné ve smyslu upgrade SW prostředků (které funkce realizují), modifikace stávajících, rozšíření o nové apod. A to bez nutnosti bezprostředního upgrade všech částí a komponent systémů, které danou funkcionalitu využívají. [10], [11]

Dohromady *vrstvy* tvoří "*zásobník komunikačních protokolů*" (*communication protocol stack*). *Zásobníky* se podle druhu určení (konstrukce konkrétního komunikačního modelu) odlišují funkcionalitou a počtem implementovaných *vrstev* a *protokolů*. [10]

Konkrétní aplikace tak může mít jen potřebné *vrstvy*. To má dopad do ekonomiky, jednoduchosti, proveditelnosti a modifikovatelnosti konstrukce; bez nákladů a problémů s implementací monolitických řešení „vše obsažných“ *protokolů*. Mj. změna fyzického média (při důsledné aplikaci této techniky) neznamena konstrukční změnu celého systému.

Několik *protokolů* může běžet na stejné *vrstvě zásobníku*. Např. *zásobník* internetových protokolů má obecně čtyři *vrstvy*, ale desítky *protokolů*. Tak Transmission Control Protocol (TCP) a User Datagram Protocol (UDP) jsou na čtvrté *vrstvě* a v principu plní stejnou funkci (komunikaci transportních entit). Ale každý za jiných provozních a technických podmínek. Jednu žádost lze odbavit TCP a druhou UDP protokolem. [10]

Celkově tak **systém může řešit řadu typově odlišných úloh** a využívat odlišné typy dat. (např. řízení: spojení, či konkrétního zařízení; přenos dat dle druhu apod.). Ke každé úloze lze využít jiný typ přenosu, případně i jiný spojovací kanál, jinou úroveň zabezpečení, různé techniky přenosu dat, zpracování stavových a chybových hlášení atd. [10]

3.2 Model otevřeného propojitelného systému (OSI)

Všeobecný standard vrstvení *protokolů* tvoří doporučení X.200 dle [11], resp. ISO/IEC 7498-1 [12], které popisují, tzv. OSI model (*open system interconnection*), otevřeností obecně je míněna zejména (budoucí) propojitelnost s technologiemi vlastní, či cizí výroby.

OSI model **není** v základu implementací zásobníku protokolů. Jde o teoretický, referenční, model, popisující obvyklé *vrstvy* komunikace a jejich funkce – metrika, s níž jsou reálné konstrukce srovnávány. Nelze tedy najít systém, či *protokol*, postavený dle X.200, ale lze najít systémy a *protokoly*, které jsou s tímto standardem částečně, či plně kompatibilní; např. *protokol* popsáný ve vztahu ke konkrétní *vrstvě*, definované X.200. Zlepšuje tak komunikaci mezi lidmi; pokud někdo říká, že jde o "protokol na třetí vrstvě", někdo jiný ví co očekávat. [10]

3.3 Vliv vrstevnaté architektury na konstrukci protokolů

Samotný návrh *protokolu* je v základu nezávislý na tom, zda konstruktér zvolí jednovrstvý nebo vícevrstvý model. Přidávání *vrstev* v zásadě znamená implementaci dalších – samo o sobě – nezávislých *protokolů*, které ve vícevrstvě komunikacím modelu plní specifické funkce a služby, resp. rozdělení monolitické architektury jedno-vrstvého řešení do více nezávislých a efektivnějších částí.

3.4 Referenční model vrstevnaté architektury dle standardu X.200

Standard se zaměřuje na přenos dat mezi obecnými zařízeními, zejména terminály, počítači atd.; zahrnuje tedy (více méně) všechna komunikující zařízení od určité úrovně složitosti. Řeší výměnu obecných dat (soubory, databáze, média, zálohování, atd.), instrukcí, synchronizačních dat řízení a výkonu distribuovaných procesů a jejich aktivit mezi *otevřenými systémy* a dále jejich integritu a zabezpečení.

Výměna informací probíhá přes fyzické médium, kterým jsou otevřené systémy propojeny.

3.4.1 ZÁKLADNÍ PRVKY VRSTEVNATÉ ARCHITEKTURY

1. **Reálný otevřený systém (*real open system*):** sestava HW a (aplikačního) SW jednoho nebo více zařízení, dle požadavků OSI na úrovni komunikace.
2. **Aplikační entita:** (*proces*) dle požadavků OSI. Zajišťuje definované funkce a představuje zdroj (a cíl) přenášených dat. Má dostupné všechny funkce

poskytované od první *vrstvy*. (Aplikace je „nad“ všemi *vrstvami*, i když má *zásobník* „aplikační“ *vrstvu*; ta je rozhraním *zásobníku* pro různé aplikace [10]).

3. **Relace** mezi aplikačními entitami otevřených systémů, na jejímž základě probíhá výměna informací mezi aplikačními entitami.
4. **Fyzické médium** spojující *otevřené systémy*, kterým se uskutečňuje komunikace. Komunikace se účastní dva a více *otevřených systémů* vzájemně médiem propojených a/nebo *otevřené systémy* propojené za pomoci převaděče.

3.4.2 VRSTEVNATOST

Otevřený systém je definován jako logická soustava vzájemně propojených a vertikálně uspořádaných, subsystémů, přičemž každý z nich plní předem dané funkce stejného typu.

Každá *vrstva* komunikuje s nejbližší vyšší nebo nižší prostřednictvím SAP (Service Access Point), který je rozhraním pro poskytování služeb vyšším vrstvám. Definují se pojmy:

- **(N)-služba** ((N)-service): služba na horním rozhraní mezi dvěma *vrstvami*; je zároveň výsledek funkcionality všech nižších *vrstev* (SAP, kde je dostupná).
- **(N)-facility(s)**: část(i) poskytované služby. *(N)-facilities* fakticky tvoří *(N)-službu*, která je zapouzdřuje, a mohou do ní být selektivně vkládány podle potřeby.
- **(N)-entita** ((N)-entity): má funkce pro přenos dat k ostatním entitám stejné *vrstvy*. Komunikuje výhradně pomocí *služeb* nižších *vrstev*. Mezi zdrojovou a cílovou entitu dané *vrstvy* může být vložena *(N)-entita*, která mezi nimi zajišťuje přenos dat. Může podporovat oba módy přenosu dle 3.4.4 a jejich *protokoly*. Každá transakce mezi dvěma a více entitami musí využívat vhodného přenosového režimu.

3.4.3 OBECNÉ VLASTNOSTI

Nemůže-li některá *(N)-entita* splnit všechny očekávané funkce [*(N)-facilities*] obsluhované *(N)-službou*, může kooperovat s jinou *(N)-entity* na stejné *vrstvě*, která danou funkci poskytuje. Kooperující *(N)-entity* se synchronizují přes nižší (N-1) nebo stejnou *vrstvu*.

Vrstvy lze dělit do *podvrstev* [*(N)-sublayer*]. Ty sdružují funkce určitého typu (*třídy*) a **lze** je v dané *vrstvě* **přemostit** (bypass). Nejméně jedna *podvrstva* vždy musí zůstat aktivní.

3.4.4 KOMUNIKACE MEZI ENTITAMI STEJNÉ VRSTVY (PEER-ENTITY) A JEJÍ REŽIMY

Obecně zajišťuje přenos dat mezi $(N+1)$ -entitami (z pohledu (N) -vrstvy) a popisuje její soustavou vlastností a funkcí:

- **(N)-connection:** pracovní propojení (*asociace*) mezi (N) -entitami. Příkaz/žádost k zřízení propojení zasílá $(N+1)$ -entita za účelem přenosu svých dat. Všechny vrstvy, vč. nejnižší, zřizují (N) -connection v režimu 1:1 nebo 1:N (více koncových bodů).
- **(N)-protokol** (komunikační): soustava komunikačních pravidel, formátování a kódování dat. Představuje všechny implementované protokoly pro danou vrstvu.

Zavádí službu **nespojovaného přenos dat** [(N) -connectionless-mode transmission]:

- Tento druh paketů se obvykle nazývá DATAGRAM (*datagramová služba*).
- **Nezaručuje doručení ani integritu** dat, neprovádí řízení přenosu. $(N+1)$ -entita nemůže vyžadovat od (N) -vrstvy žádné spolehlivé služby a (N) -vrstva (implicitně) nepředává zpět žádné informace o průběhu přenosu. Před zahájením přenosu nejsou alokovány žádné přenosové zdroje, ani parametry přenosu; kromě adresy cílové (N) -entity a některých parametrů QoS, nutných k doručení každého paketu síti nezávisle na ostatních.
- **Není definována minimální úroveň QoS.** Pokud jsou parametry QoS definovány, vycházejí z výsledků minulých relací a jsou nesený každým *datagramem* nezávisle.
- Služba umožňuje pracovat s momentálním výkonem a spolehlivostí přenosového kanálu/média. Každý paket může být zaslán nezávisle množině příjemců.

Zavádí *spojově orientovaný přenos dat* [(N) -connection-mode transmission] (**spojovaná služba**): Přenos dat v paketově orientované síti využívá kompletní transakce přenosu každého paketu nebo jejich skupiny. Umožňuje detekci ztrát (spojení, paketů), podvrhů, zbloudilých paketů a duplikátů a plnou implementaci QoS, která zaručuje kvalitu a zpoždění přenosu. Pracuje ve třech fázích, které jsou zároveň základními službami vrstvy:

- **Přístupová fáze:** zdrojový systém navazuje spojení s cílovým (*connection establishment*); vytvoří dvou nebo vícebodové spojení, charakterizované vzájemnou akceptací a dodržováním (technických) podmínek přenosu. Entity přenosu vyjednávají podmínky spojení, alokaci zdrojů a tento *kontext* je uchován po dobu existence příslušné transakce. Viz také: 3.4.8, zřízení spojení.

- Fáze přenosu dat, udržování a řízení spojení. Uživatelská nebo nezávislá servisní data obsahují informace pro řízení spojení, optimalizuje jejich přenos (kterým režijním datům se lze vyhnout). Poskytuje *kontext spojení*, (přenášené datové jednotky (DU) spolu logický souvisí), zajišťuje řízení toku transakcí a doručení DU ve správném pořadí.
- Fáze ukončení spojení (transakce) (*connection release*); viz 3.4.8, uvolnění spojení.

Protokoly spojově orientovaných služeb jsou implementovány na 1, 2, a 3. *vrstvě* OSI.

Služba je vhodná pro aplikace vyžadující dlouhodobé, stabilní interakce mezi entitami (propojení terminálů vzdálených počítačů, přenos souborů, streamově orientované služby a dlouhodobé připojení vzdálených systémů).

Na rozhraní (*N*)-SAP mezi dvěma *vrstvami* může docházet ke konverzi přenosových módů – mezi vyšší a nižší *vrstvou* dojde ke změně módu ze spojovaného na datagramový a opačně. Pro *spojovaný režim* musí být zajištěna kapacita a přenos dat pro řízení přenosových transakcí; pak přenos spojované služby je možné provádět pomocí více (*N*)-*spojení* ((*N*)-*connection*) datagramové služby a opačně.

Aby bylo možné přenos dat (bez ohledu na zvolený režim přenosu) uskutečnit, je nezbytné, aby zdrojové (*N*)-*entitě* byly předem známy následující parametry: adresa a dostupnost cílové (*N*)-*entity*, *protokol* přenosu, kvalita spojení (QoS).

3.4.5 IDENTIFIKACE

SAP jako rozhraní k propojení (*N*)-*entit* mezi *vrstvami* identifikuje adresu, unikátní z pohledu (*N*)-SAP. Překlad a identifikaci adresy zajišťuje (*N*)-*entita*. Pro entity jiných *vrstev* může být adresa používaná (*N*)-*entitou* nečitelná. Proto jsou celé DU vyšších *vrstev* vkládány do DU *vrstev* nižších tak, že po přenesení (*N-1*)-*vrstva* příjemce poskytne následující (*N*)-*vrstvě* celou DU tak, jak jí předal odesílatel.

Aby (*N+1*)-*entita* mohla navázat spojení s jinou, přidělí (*N*)-*entita* každé (*N+1*)-*entitě* unikátní identifikátor koncového bodu, vytvořený z adresy (*N*)-SAP a přípony koncového bodu příslušného (*N*)-*spojení*, kterým probíhá přenos dat. V případě přenosu mezi více koncovými body se použije identifikátor koncových bodů násobného spojení, který je unikátní v rámci daného (*N*)-*spojení* a identifikuje koncové body, které data přijímají. (Např. TCP soket se skládá z IP adresy koncového bodu a portu na kterém naslouchá příslušná služba (př.: 192.168.1.2:8080)).

3.4.6 KOMUNIKACE MEZI ENTITAMI SOUSEDNÍCH VRSTEV

Probíhá v rámci téhož subsystému, (N) -SAP na rozhraní *vrstev* (vertikálně). Každá entita může být v jednom okamžiku připojena k více (N) -SAP, ale v jednom okamžiku může být (N) -SAP připojen jen k jedné (N) -entitě a $(N+1)$ -entitě.

(N) -SAP smí podporovat (N) -službu v obou komunikačních režimech, resp. jejich kombinaci. Pokud $(N+1)$ -entita používá více spojení prostřednictvím jednoho, či více (N) -SAP, pak využívá (N) -služby v nespojovaném režimu.

3.4.7 DATOVÉ JEDNOTKY

Nejsou omezení vnitřní struktury nebo velikosti DU *vrstvy*. Jsou definovány pouze typy a povinné části DU. Specifické deklarace a implementace *protokolů* a služeb každé *vrstvy* tato omezení mít může.

- **DU (N) -protokolu** (PDU – *protocol data unit*): obsah interpretuje (N) -služba dle deklarace (N) -protokolu. Přenos PDU mezi (N) -entitami je transparentní a použitím $(N-1)$ -služby, která PDU mapuje na $(N-1)$ -PDU tak, že (N) -SDU, jako produkt zpracování přenášených dat (N) -službou, doplní řídicí informací (N) -protokolu, (koordinace komunikace (N) -entit). K přístupu k $(N-1)$ -službě využívá $(N-1)$ -SAP.
- **DU (N) -služby** (SDU – *service data unit*): produkt zpracování přenášených dat (N) -službou. Nese informaci k udržení identity mezi (N) -entitami.
- **Spěšná DU (N) -služby** (ExSDU): má omezenou velikost a zaručenou prioritou doručení a zpracování. Související služba se používá pro signalizaci a přerušení.
- **PCI** (protocol control information): data vyměňovaná (N) -entitami k vzájemné koordinaci společné činnosti (řízení spojení).

3.4.8 ČINNOST A FUNKCE VRSTEV

Standard definuje základní společné funkce, které by (N) -služby měly poskytovat z hlediska záruky propojitelnosti *otevřených systémů*.

- **Výběr a identifikace *protokolu***: (N) -vrstva ((N) -layer) může používat více *protokolů*, které pak mají unikátními identifikátory. Standard nedovoluje přenos ID *protokolu* jako součást PCI. Protokoly jsou identifikovány (N) -službou, použitím (N) -adres. Přesná specifikace odkazuje do ITU-R X.650, resp. ISO 7498-3.

- **Identifikace verze *protokolu*:** ID nese PCI a umožňuje přesné sladění rozsahu a formátu přenášených dat, resp. volání odpovídající verze *protokolu* na specifickém *(N)-spojení*. *Datagramové služby* nastavení neprovádí, předpokládá buď shodnou verzi *protokolu* na obou stranách, nebo je ID verze v PDU. Funkce umožňuje vývoj *protokolu* bez globálních následků (ve smyslu povinného upgrade na všech implementacích v okamžiku uvedení verze do provozu). Proveďte se upgrade jen systémů, kde je třeba nových funkcí *protokolu*. Verze *protokolů* jsou obecně kompatibilní, je-li zachován způsob sdělování, kódování, identifikace verze *protokolu* a chování z pohledu *(N)-entity*.
- **Zřízení spojení:** mezi *(N)-entitami*; spojení se zřídí vertikálně mezi všemi entitami na nižších *vrstvách*. Pokud některá *vrstva* nemá odpovídající *služby*, spojení se nezřídí. Spojení na všech potřebných *vrstvách* může být navázáno postupně. Spojení mezi *(N)-entitami* mohou zříditi: (a) k tomu určená *(N)-spojení* s dedikovanými *protokoly*; tyto relace mohou být následně využity k řízení spojení; nebo (b) je komunikace k navázání spojení přenášena přímo s uživatelskými daty *(N)-vrstvou*. Volba přístupu záleží na technických podmínkách komunikace. Měla by být zvolena efektivnější a spolehlivější varianta z obou.
- **Uvolnění spojení:** zahajuje *(N+1)-entita* nebo *(N)-entita* jako důsledek detekované chyby na *(N)-vrstvě* nebo nižších, resp. *(N)-entita* na žádost *(N+1)-entity*. Entity, které zahájily uvolnění, musí provést nezbytnou komunikaci. Proces uvolnění *(N)-spojení* nemusí uvolnit *(N+1)-spojení*; to může být v domluveném čase obnoveno, či nahrazeno. Při okamžitém uvolnění může proces skončit ztrátou uživatelských dat; nebo se po zahájení sekvence ukončení čeká na dokončení přenosu uživatelských dat a teprve potom se spojení ukončí. Komunikační sekvence k uvolnění smí být přenášena společně s uživatelskými daty.
- **Pozastavení a pokračování spojení:** *(N+1)-vrstva* vyvolá požadavek ukončení *(N)-spojení*, přičemž *(N+1)-spojení* je zachováno (viz výše). Funkce k efektivnímu využití přenosového média, např. při plánované nečinnosti *(N+1)-entit*. Na žádost *(N+1)-entita* spojení na *(N)-vrstvě* (a nižších) obnoví a pokračuje v komunikaci.
- **Multiplexing:** optimalizace – násobné využití *(N)-spojení*; Provádí identifikaci všech PDU, proti zkřížení uživatelských dat, řízení datového toku *(N+1)-vrstvy* k efektivnímu sdílení kapacity *(N)-spojení*.

- **Demultiplexing:** společné využití více (N) -spojení jednou (N) -entitou k zvýšení výkonu, spolehlivosti a odolnosti komunikace. Ekonomická optimalizace provozu využitím více levnějších spojení, než by byla cena jednoho požadovaného. Provádí plánování využití násobného (N) -spojení a správné řazení PDU po příjmu.
- **Řízení toku dat:** přizpůsobuje rychlost odesílání PDU mezi (N) -entitami. Je podporováno v obou režimech přenosu. Je definováno protokolem a řídicí informace nese PCI. Dále je implementováno na rozhraní (N) -služby, kde řídí rychlost předávání DU mezi (N) -službou a $(N-1)$ -službou.
- **Spěšný přenos dat** (spojované služby): lze chápat jako spojení dvěma subkanály. Jeden je určen pro spěšná data. Díky malému objemu dat používá zjednodušený mechanismus řízení spojení. Priorita je definována pro (N) -spojení (N) -vrstvy; v jiných (N) -vrstvách nemusí mít stejnou prioritu. Každá (N) -vrstva má nezávislý mechanismus řízení toku ExSDU, který nelze obejít. Proto se nedoporučuje:
 - mapování (N) -ExSDU do nižších vrstev, resp. jejich služeb.
 - ExSDU má být zcela řízena (N) -funkcí a ta má využívat jen nezbytné služby nižších vrstev;
 - velikost ExSDU se má volit tak, aby nebylo potřeba segmentování.
 - Nemá se používat pokud $(N-1)$ -vrstva multiplexuje.
 - Nesmí se používat jako kanál pro trvalý přenos s odlišnou prioritou.

U nespojovaných služeb lze obdobného efektu (bez záruky) dosáhnout nastavením odlišných QoS (priorita, dovolené zpoždění).

- **Segmentování:** mapování větší SDU (N) -služby do více menších PDU jiné vrstvy (přizpůsobení potřebám protokolů na různých vrstvách). Segmenty jsou opatřeny ID k zachování jejich identity a do PCI mohou být uloženy další řídicí údaje, aby na konci přenosu mohla (N) -entita znovu sestavit původní PDU.
- **Blokování:** opačný proces k segmentování (menší SDU jsou mapovány do jedné větší PDU). Dostupná jen pro spojované služby.
- **Řetězení:** více (N) -PDU je spojeno do jediné PDU jiné vrstvy.
- **Řazení:** řadí PDU doručené $(N-1)$ -službou do správného pořadí; $(N-1)$ -služba principiálně nezaručuje doručení DU v pořadí, jak byly odeslány. Vyžaduje dodatečné řídicí informace v PDU. V nespojované službě realizována nepřímo.

- **Potvrzování:** (spojovaná služba): využívána (*N*)-*entitami* k zvýšení spolehlivosti doručení, než jaké poskytuje (*N-1*)-*vrstva*. PDU mají unikátní ID a o doručení je zpět odeslána informace. Řídicí informace jsou v PCI a mohou být využity k odstranění duplicitních PDU, řízení segmentování a řazení. V nespojovaném režimu lze potvrzovat pro jednotky (*N*)-PDU, nikoliv pro (*N*)-SDU.
- **Detekce chyb:** PDU jsou doplněny o záznam pro opravu dat, uložený v PCI. U nespojované služby nelze zajistit ochranu proti ztrátě celých DU.
- **Reset** (spojovaná služba): nastaví relační (*N*)-*entity* do předem známého stavu. Může dojít ke ztrátě a duplikaci dat. Může vyžadovat komunikaci dotčených (*N*)-*služeb* a (*N*)-*entit*, resp. uložení řídicích dat do PCI.
- **Směrování:** komunikaci (*N*)-*entit* zprostředkovává řetězec mezilehlých (*N*)-*entit*; tato skutečnost zůstává skryta entitám na ostatních *vrstvách*. Směrovací zařízení může mít jen počet *vrstev* potřebný k výkonu své funkce. Převod se uskutečňuje uvnitř (*N*)-*vrstvy*. Standard definuje (*N*)-*převáděcí zařízení*, k propojování různých sítí nebo jejich paralelizaci. (*N*)-*převáděcí zařízení* může být libovolný *otevřený systém* s odpovídající funkcionalitou. Dále se služba využívá k dosažení (*N+1*)-*entity*, má-li tato entita více společných SAP s jednou a více (*N*)-*entitami*.
- **Jakost služby (Quality of Service – QoS):** (ne/spojované služby). Zahrnuje řadu parametrů, specifických každé *vrstvě* na základě technických funkcí a podmínek, za kterých je *vrstva* plní. Standard parametry uvádí jako příkladové, tzn. je na konkrétní konstrukci *protokolu*, resp. provozovateli spojení, které konkrétní parametry zvolí a jak je zkombinuje. Při aplikaci na spojované služby jsou parametry QoS stanoveny (vyjednány) ve fázi navazování spojení komunikujícími entitami. U nespojovaných služeb jsou parametry definovány chováním při předcházejícím jednorázovém přenosu. Kromě parametrů stran efektivity přenosu DU je definována ochrana před neoprávněným přístupem. Základním dokumentem, který rozpracovává QoS je doporučení ITU-T E.800, které definuje základní pojmy. Řada dalších doporučení řeší konkrétní aplikace.

3.5 Obecné zásady při určování vrstev

Standard uvádí teoretické zásady návrhu vrstevnatosti, ze kterého mj. vyplývá počet sedmi *vrstev* modelu OSI. Zásady jsou abstraktní a lze je využít jako základ, při modelování

specifických komunikačních systémů, resp. OSI model optimalizovat vynecháním nepotřebných *vrstev*. Všechny zásady se dotýkají způsobu, jak stanovit hranice *vrstev*:

- Vytvářet minimální potřebný počet *vrstev*
- Vytvářet co nejjednodušší *protokol*
- Všechny *vrstvy* kromě aplikační, se podílejí na zdokonalování komunikačních služeb, přičemž (dosažený) stupeň zdokonalení identifikuje hranici mezi dvěma sousedícími *vrstvami*, tedy v úrovni (N) -SAP, resp. (N) -služba.
- Hranice má být zároveň zvolena tak, aby počet interakcí přes hranici byl minimální
- Funkce a chování *vrstev* jsou dány deklarací *protokolů* každé *vrstvy*
- Vytvořit zvláštní *vrstvy* pro funkce zjevně odlišné procesně a/nebo technicky
- Podobné funkce soustředit do jedné *vrstvy*, aby její funkce a *protokoly* byly snadno a široce modifikovatelné, protože takový zásah pak nezasahuje přes hranice *vrstvy*
- Stanovit hranice tam, kde se mění morfologie, sémantika a syntaxe dat.
- Shodně vytvářet i *podvrstvy* v rámci *vrstev*. Smyslem podvrstev je optimalizace (minimalizace) počtu a rozsahu komunikačních funkcí *vrstvy*.

4 TESTOVÁNÍ KOMUNIKAČNÍCH PROTOKOLŮ

Zdroj: pokud není v textu konkrétně uvedeno jinak, vychází celá kap. 4 z [1], kap. 5.

Testování *protokolů* a jejich implementace je nepominutelná fáze jejich konstrukce. Následuje po implementaci konstrukčního návrhu a primárním cílem je ověřit implementaci ve vyšším programovacím jazyku a model *protokolu*.

Testování vychází z praktik SW inženýrství a může zohledňovat technické normy a specifikace, normy manažerské a řízení kvality – ve smyslu stanovení standardů kvality, spolehlivosti, dokazování a ověřování shody zkoušek atd.

Jako ucelený proces přímo ovlivňuje kvalitu výsledného produktu a současně konzumuje nezanedbatelné množství zdrojů. Proto představuje obor, který soustavně vyvíjí nové techniky a přístupy s cílem maximalizovat efektivitu.

4.1 Základní předpoklady

V rozsáhlých projektech se doporučuje personálně oddělit vývojové, implementační a testovací týmy. Vedle organizační a kapacitních otázek je smyslem tohoto opatření omezit

rutinní chování personálu, založené na subjektivních předpokladech (které by mohly vést k výskytu latentních chyb a slabých míst) získaných prací na jedné z uvedených etap vývoje. Proti tomu stojí techniky extrémního programování, jak je popsáno dále.

Testovací scénáře a metodologie musí úzce vycházet z projektovaného chování *protokolu*. Důvodem je, že samotná analýza *protokolu* zahrnuje úvahy o vstupu a výstupu (scénáře vývoje výsledku). Testovací scénář zahrnuje všechny (vybrané) testovací případy, tedy **množinu párů chování vstup-výstup**. Základní scénáře mají vznikat již při definici modelu požadavků a při tvorbě vztahových diagramů.

Testování se provádí sadou níže popsaných testů. Ta je zpravidla navržena v jazyku TTCN (Tree and Tabular Combined Notation; ISO/IEC 9646, resp. ITU-T X.290). Samotná testovací sada je realizována vyšším programovacím jazykem (Java, C ++ apod.).

Případné chyby v testovací sadě se ověřují v rámci testů; **neexistuje nezávislá kontrola testovací sady** - vždy se kontroluje správnost testované implementace a test současně.

Počet po testování zbývajících chyb ve finálním produktu je hlavní metrika kvality SW a schopnosti výrobce jej dodat. Další důležitou metrikou jsou testy pokrytí, které řeší rozsah testování z hlediska počtu testovaných variací SW cest, kombinace proměnných atd.

4.2 Testování podle konvenčního SW inženýrství

4.2.1 TESTOVÁNÍ MODULŮ (*UNIT TESTING*)

Testování SW modulů před jejich integrací do výsledného produktu s cílem ověřit jejich správnost. Modul je typicky jedna *třída* v samostatné knihovně, která implementuje jednoduchý *protokol* nebo část složitějšího). Obvykle jde o sadu testovacích scénářů (algoritmů a podmínek jejich provedení), které kontrolují individuální změny *stavů* konečného automatu (KA) a složitější transakce KA (řada *stavových přechodů* KA). Mají pokrývat hraniční podmínky a situace nezvladatelné cílovým SW.

Podle metodiky extrémního programování testy píše před, nebo alespoň v průběhu implementace cílového systému autor modulu. Změnou role z programátora na testera se zaměří na rozhraní modulu a dostává jasnější obraz o službách, které modul v cílovém SW poskytuje. Dále se tvorbou modulových testů více seznámí s cílovou implementací a dostane okamžitou zpětnou vazbu - chybu je snadné odhalit v rozsahu konkrétního testu. Tím celkově zvyšuje efektivitu a produktivitu své práce.

Testy se provádějí kompletně, automaticky a mnohonásobně v průběhu implementace testovaného modulu a dále, při každé změně jeho zdrojového kódu – tzv. **regresní testování**. To mj. umožňuje a podporuje alternativní přístup k vývoji a tím i hledání optimálního řešení. Důvodem násobného opakování testů je statistické ověření a odhalení případných skrytých chyb testovacího framework.

Klíčovými předpoklady úspěšného testování jsou:

- Správný rámec testování
- Vyloučit lidský zásah do testu (vstupních dat, kontroly průběhu testu, záznamu výsledků, selhání a přerušení testů a jejich důvodů). Testovací scénáře a zdrojový kód modulu se nesmí po dobu provádění testů měnit (ani příkazy pro uložení výstupu funkcí do souboru). Lze psát nové třídy děděné ze základní třídy modulu v rámci funkcionality, kterou poskytuje testovací framework.
- Hierarchie testovacích scénářů s možností rozšiřování (hlavní a podřízené testy).
- Hromadné provedení sady testů definované v rámci dané hierarchie
- Podrobný protokol každého testu.

4.2.2 INTEGRAČNÍ TESTY

Testování modulů spojených do výsledného produktu. Má několik přístupů provádění:

- **Top-down:** testován je vždy modul hierarchicky nad těmi, které modul volá;
- **Bottom-up:** opačné k top-down;
- **Sandwich:** moduly se testují podle jejich logické funkce v modelu (I/O, logické, hlavní apod.);
- **Big-bang:** celá implementace je testována najednou. Podrobnější informace o jednotlivých přístupech, jejich výhodách a nevýhodách a uplatnění jsou v [15].

Metodika předpokládá, že nejvíce chyb je na rozhraní modulů –vzájemná nekompatibilita funkcí, proměnných a ve sdílení paměti, a že některé moduly úspěšně prošly testy, jsou připraveny pro další testování, zatímco zbytek neprošel. [15], [1], kap. 5.

Zavádí se náhrada modulů, které neprošly testy nebo jsou nedostupné z jiného důvodu, tzv. simulátory; modelují chování daného modulu očekávané na jeho rozhraní (vstup a výstup). Základní dva druhy simulátorů jsou:

- **Drivers:** aktivní imitátor, generuje vstupní *zprávy* pro testované *objekty*

- **Stub:** pasivní imitátor, přijímá výstupní *zprávy* od testovaných *objektů* a posílá odpovědi, které testované objekty očekávají.

Protokoly jsou vhodné k integračním testům. Jejich vnitřní struktura i vnější kombinace jsou organizovány ve *vrstvách* a mají transparentní rozhraní. Komunikace *protokolů* (a jejich vnitřních modulů) je založena na výměně *zpráv*. Dále se *protokoly* obvykle implementují od spodní *vrstvy* k horní. Simulovat testovací prostředí je pak snadné; *drivery* a *stuby* prostě vyměňují *zprávy* s testovanými *objekty*, resp. simulují pouze část vyšší *vrstvy zásobníku protokolů*.

4.2.3 TESTY SHODY (CONFORMANCE TESTING)

Též black box testing – testy nezávislé na vnitřní struktuře implementace.

První fáze akceptačního testování. Tester neřeší vnitřní strukturu a chování SW. Zjišťuje, zda vnější chování splňuje určené specifikace.

Obvykle se testy specifikují TTCN scénáři a kontroluje se odpovídající (očekávaná nebo zakázaná) reakce funkcí implementace na určité *události* (a jejich kombinace). Testy nemají být složitější, než je nezbytně třeba a funkce mají být zkoušeny v očekávaném provozním rozsahu. Složité kombinační situace jsou účelem zátěžových testů.

V dokumentovaných případech se shoda testuje podle zvolených standardů a doporučení (typicky autorit jako IETF, ISO, ITU-T, ETSI, IEEE apod.). Ty vedle funkcionálních požadavků *protokolů* definují i požadavky jejich zkoušek a ověřování. Jejich součástí může být kompletní předpis testu (definice a rozsah zkoušených funkcí a role jejich uplatnění) nebo abstraktní popis testu, požadavky na provedení implementace a na schopnosti producenta, definice testovacího prostředí a aplikační scénáře (podmínky technické aplikace testované implementace).

V případě, že jde o veřejně dostupné výrobky, mají testy shody zajišťovat příslušné autorizované osoby. Jde zejména o to, aby tester disponoval odpovídajícím technickým vybavením. Odborná úroveň a zkušenosti zahrnují provádění měření a používání měřicí techniky, schopnost důvěryhodné implementace požadovaného testovacího framework a schopnost legislativního vyjadřování.

Menší subjekty s méně sofistikovaným vybavením, které ale mají dostatek zkušeností lze využít pro privátní testování, kde není požadována legislativní deklarace shody nebo jako

před-testery, kteří odhalí, chyby v testované implementaci a tím významně snižují celkové náklady formálních testů. Pro tento účel je dostupná řada testovacích framework – viz 4.4.

Přehled možného procesu prokazování shody popisuje ISO/IEC 9646-1, resp. ČSN EN ISO/IEC 9646-1. Krátký přehled poskytuje [18].

4.2.4 ZÁTĚŽOVÉ TESTY (LOAD TESTING)

Provádí se v simulovaném prostředí (laboratoři) za použití zátěžového generátoru, který nabízí výběr určených testovacích scénářů a jejich parametrů (počet požadovaných zdrojů, doba trvání jednotlivých fází komunikace, atd.). Následně na základě reakcí testovaného systému vyhodnocuje a ověřuje jeho chování.

Primárně kontrolují deklarované komunikační schopnosti produktu (např. míru zahozených žádostí, ztracených DU, chování za stavu extrémně nízké míry komunikačních požadavků, tj. kontrola udržitelnosti dlouhodobých připojení a za stavu extrémně vysoké míry komunikačních požadavků, tj. kontrola mechanismů ochrana přetížení, apod.).

4.2.5 PROVOZNÍ TESTY (IN-FIELD TESTING):

Poslední fáze přejímacího řízení – experimentální nasazení produktu na omezenou dobu (např. tři měsíce).. Cílem je zjistit, lokalizovat a eliminovat chyby, které odhalí (jen) reálný provoz, protože je není možné simulovat v laboratoři. Provoz se zaznamenává do logů (soubor protokolu) a analyzuje se. Součástí tohoto testu je i programová simulace výjimečných nebo neobvyklých stavů a komunikačních transakcí.

4.3 Přístupy cleanroom metodologie

Cleanroom metodologie představuje trend přístupu i pro konstrukci *protokolů*, resp. SW inženýrství obecně. Popsané techniky se dotýkají oblastí formální a věcné kontroly modelu implementace, modelování operačního modelu, automatického generování sestavy případových testů a statistického testování spolehlivosti. *Další literatura: např. [13].*

4.3.1 FORMÁLNÍ TESTY AUTOMATICKÉHO DOKAZOVÁNÍ

Metodika založená na teorému automatického dokazování (*automated theorem proving* – ATP) a na predikátové logice 1. řádu (*další literatura – např. [14]*). Používá se pro formální kontrolu specifikace *protokolu* a jeho implementace.

Základem metodiky je axiomální (matematický) model očekávaného chování KA vyjádřený soustavou jednoznačných vzorců predikátové logiky. Každý vzorec modeluje jeden stavový přechod KA (větev). Definují se:

- **Kontrolní predikát $A(Nn)$:** řídí aktivitu KA; A je automat (entita), Nn značí konkrétní KA (**Globální kontrolní predikát:** zapíná nebo vypíná konkrétní KA),
- **Predikát řízení změny stavu $T(Mn)$:** řídí aktivitu každého jednoho konkrétního stavu KA; T značí logiku řízení přechodu stavu a Mn konkrétní stav KA.
- **Konkrétní stav KA:** $S(n)$
- **Konkrétní vstup KA:** $I(n)$
- **Konkrétní výstup KA:** $O(n)$

Pokud ve vzorcích nejsou zahrnuty prvky $A(Nn)$ nebo $T(Mn)$ je (vzhledem k modelu protokolu) vypnutý buď celý konkrétní KA, nebo jeho konkrétní stavy. Tabulka 8 uvádí příklady axiomatických vzorců (dále vzorců).

Touto teorií se definuje test každého KA (definovaného v modelu protokolu), jako věta o průběhu konkrétního stavu konkrétního KA. Říká, že pro danou sérii vstupů KA $\{I1...In\}$, tento KA provádí sérii stavových přechodů $\{S1...Sn\}$, které na jeho výstupech $\{O1...On\}$ produkují konkrétní stavy. (Věta 8 v Tabulka 8).

Reálné *protokoly* obvykle nesestávají z izolovaných KA, ale z funkčních bloků (skupin KA – *stavového stroje*), propojených komunikačními kanály, kterými si zasílají *zprávy (události)*. *Stavový stroj* je popsán jako sjednocení množin *vzorců* dílčích KA. Teoretická definice testu *stavového stroje* je zobecněním teoretického modelu KA. Levá strana *vzorce* obsahuje řídicí predikát (pokud existuje) a vysílané *signály*. Pravá strana obsahuje výsledné *stavy* a výsledné *signály*.

Stavový stroj představuje procesy *protokolu* a komunikační kanál jednu a více signalizačních cest. Propojení lze modelovat různě:

- tradiční dekompozicí,
- jako chaotický systém (každý komunikuje s každým, např. bezdrátové sítě),
- jako různé topologické systémy (hvězda, kruh, sběrnice) apod.

K tomu se zavádí další prvky axiomální rovnice:

- **Akt zaslání signálu $Sig(SigN)$:** Sig značí predikát, $SigN$ jeho uživatelské jméno

- **Akt zaslání konkrétního signálu konkrétní signalizační cestou $SoP(SigN, Pm)$:**
 SoP je predikát, $SigN$ viz výše a Pm je jméno konkrétní signalizační cesty.

Následné dokazování teorémů se provádí automaticky. [1], kap. 5 odkazuje na systém THEO. Existuje řada dalších komerčních a volně dostupných nástrojů. Základní přehled je dostupný např. z: http://en.wikipedia.org/wiki/Automated_theorem_proving.

Výstupem nástroje je soubor protokolu s průběhem výkonu operací a změnou *stavů* KA.

Z hlediska provedení operací je model KA časově nezávislý, tedy neexistuje časové razítko *události* v logu modelu a operace prováděné (s) jednotlivými operandy, lze vykonat (tedy i zapsat) v libovolném pořadí, protože operátor ‘&’ představuje komutativní operaci.

Teoretický test tak lze nahlížet ze dvou hledisek:

- jeho výsledek je platný navždy,
- všechny operace se staly v jediném časovém okamžiku.

Časová nezávislost *vzorců* je jejich klíčová prokazovací síla. Odhaluje kritické následky (v praxi nepravděpodobného) souběhu dvou a více *událostí* v čase, resp. neočekávaného (poruchového) opakování vstupních *signálů* a *stavů*; protože každý vstup lze ve *vzorci* definovat libovolně krát po sobě. To redukuje nárok na testovací kapacity.

Pro testování *protokolů* se tento nástroj používá tak, že jsou stanoveny dvě role testerů. První zapíše *vzorci* a druhá je přepíše do testovacího nástroje a vykoná testy. Pokud se v teoretickém testu vyskytne chyba – nelze nalézt důkaz platnosti *vzorci* – je ve *vzorci* nebo jeho modelu chyba. Pokud se nepodaří nalézt triviální (syntaktickou) chybu ve *vzorci* nebo zápisu modelu, jde o vnitřní chybu modelu *protokolu*.

Jednou z cest jak se takovým chybám vyvarovat, a také jak je následně hledat je zkracování (optimalizace) *vzorců* na minimum. Dále se v těchto případech užívá bloková analýza (postupné prokazování platnosti, např. práci s kontrolními predikáty a jednotlivými vstupy) a další techniky.

Metoda ATP se používá k automatickému generování sestav teoretických testů. Pokud o nějakém *vzorci* předpokládáme, že je správný, vybere se některý ze *signálů* (*událostí*) na levé straně *vzorci*. ATP nástroj vrátí sadu možných vstupních výsledků, ze které se hledá ten, který je důkazem *vzorci*. Pokud existuje, představuje důkaz pro daný *signál*. Pokud některé vstupní *signály* generují jen vnitřní *přechod stavu* KA, pravá strana *vzorci* zůstane prázdná. Opakováním tohoto postupu lze vygenerovat teoretický test libovolné délky.

Generování testů je efektivnější, pokud při jejich návrhu lze sledovat a respektovat vnitřní strukturu KA, ne jen vycházet z jeho vnějšího chování (black box testování).

V případě, že testovací sada již existuje (je formálně deklarována držitelem *protokolu*, např. ETSI), přepíše se a přeloží do *vzorců* a následně se provede formální test shody *vzorce* modelu se standardem (deklarací *protokolu*).

Lze rovněž použít nástroj revizního inženýrství k extrakci *vzorců protokolu* ze zdrojového kódu modelu, případně z jeho dostupných logů. ATP se dále používá k modelování grafu kompletního rozsahu přechodových *stavů* KA.

Tabulka 8: Příklady axiomálních vzorců KA a stavového stroje:

#	Axiomální vzorec	Komentář
1.	State(INITIAL)	Nepovinný úvodní vzorec (<i>výchozí stav</i> KA). Pokud je nutné test spouštět z různých <i>výchozích stavů</i> , je <i>výchozí stav</i> definován na levé straně <i>vzorce</i> (viz níže). <i>State</i> je predikát a <i>INITIAL</i> <i>výchozího stavu</i> KA
2.	$\{S(X) \ \& \ I(T)\} \Rightarrow \{S(Y) \ \& \ O(R)\}$	<i>Vzorec</i> časově nezávislé změny <i>stavu</i> X do <i>stavu</i> Y, spouštěn vstupem T a produkující výstup R. $\{X, Y, T, R\}$ jsou konstanty; definují <i>výchozí</i> a <i>finální stav</i> a konkrétní vstup a výstup KA.
3.	$\{S(X) \ \& \ I(T)\} \Rightarrow \{S(Y) \ \& \ O(R_1) \ \& \dots \ \& \ O(R_n)\}$	<i>Přechod stavu</i> generuje více než jeden výstup
4.	$\{A(N) \ \& \ T(M) \ \& \ I(I_1) \ \& \dots \ \& \ I(I_n)\} \Rightarrow \{O(O_1) \ \& \dots \ \& \ O(O_n) \ \& \ S(S_1) \ \& \dots \ \& \ S(S_n)\}$	Kompletní vzorec s řídicím a kontrolním predikátem, násobné I/O a <i>stavy</i> KA.
5.	$\{A(N) \ \& \ T(M) \ \& \ I(I_1) \ \& \dots \ \& \ I(I_n)\} \Rightarrow \{O(O_1) \ \& \dots \ \& \ O(O_n) \ \& \ S(S_n)\}$	Kompletní vzorec s řídicím a kontrolním predikátem, násobné I/O a <i>stavy</i> KA a finální stav KA.
6.	$\{S(X) \ \& \ Sig(P)\} \Rightarrow \{S(Y) \ \& \ Sig(Q)\}$	Změna <i>stavu</i> X do Y, vyvolaná přijetím <i>signálu</i> P, vyvolávající odeslání <i>signálu</i> Q. Ovládání <i>signálů</i> se děje bez ohledu na cestu.
7.	$\{S(X) \ \& \ EoP(E_1, P_1)\} \Rightarrow \{S(Y) \ \& \ SoP(E_2, P_2)\}$	Změna <i>stavu</i> X do Y, vyvolaná přijetím <i>signálu</i> E1, zaslaného cestou P1, vyvolávající odeslání <i>signálu</i> E2, cestou P2.
8.	$\{Sig(A)\} \Rightarrow [Sig(B) \ \& \ Sig(C) \ \& \ Sig(D) \ \& \ S(X) \ \& \ S(Y)]$	Změna <i>stavu stavového stroje</i> spouštěná vstupním <i>signálem</i> A (generuje <i>signály</i> {A,B,C} a <i>stavy</i> {X,Y}).

4.3.2 STATISTICKÉ TESTY SPOLEHLIVOSTI

Statistical usage testing (též behaviorální testování). V současnosti hlavní metoda hodnocení kvality průmyslových a embedded systémů. Ideou je testovat určitý produkt (nebo jeho část) za podmínek očekávaných v reálném provozu.

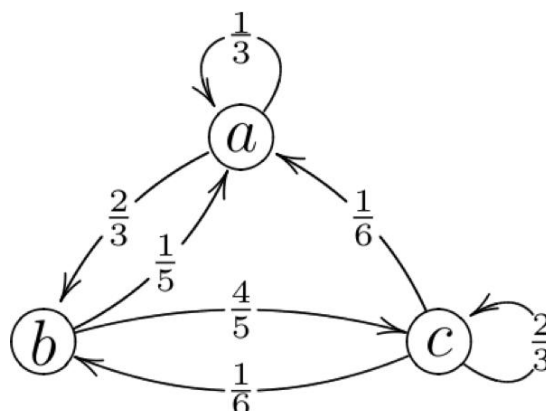
Úvahy o kvalitě (spolehlivost) výsledného produktu vychází z metrik kvality tradičního SW inženýrství, které uvažuje s množstvím chyb zbývajícím po testech a rozsahem pokrytí testované implementace případovými testy. Tato teorie však není záruka spolehlivosti výsledného produktu, neboť neuvažuje s pravděpodobností výskytu jediné zbývající chyby; [1], kap. 5 v tomto ohledu uvádí (ad absurdum) jedinou zbývající chybu v implementaci, která vždy při startu způsobí její pád. V implementaci sice zbývá jediná chyba, ale ta kompletně znemožňuje její projektované užití. Analogicky se tato situace přenáší i do prostředí inženýringu *protokolů*.

Každý netriviální výrobek vyžaduje k ověření správné funkčnosti testy v řádu desítek a stovek tisíc případů. Cílem této metodiky je vybrat ty provozní *stavy* a *události*, které běžně převažují. To má pozitivní dopad na spolehlivé otestování klíčových funkcí a vlastností SW, dovoluje omezit rozsah sady testů a tím i nutné prostředky, resp. využít je na otestování prioritních vlastností a funkcí.

Měří se procentuální spolehlivost implementace jako poměr celkového počtu provedených testů k absolutnímu počtu testů vyplývajících pro danou implementaci. Měření je nepřímé proto, že přímé měření není často reálné. Počítá se počet případových testů nutných k dosažení předpokládané spolehlivosti. Zbytek padá do kategorie zbytkového, resp. akceptovatelného rizika a je mírou schopnosti producenta SW jej dodat.

Deklarace provozních (testovaných) podmínek je dána ***operačním profilem*** testovaného produktu. *Operační profil* je v podstatě *stavový stroje* s definovanou pravděpodobností *přechodu stavů*. Pravděpodobnost výskytu určitého *stavu* (*události*) je podíl počtu reálných výskytů k absolutnímu počtu možných výskytů události.

Z matematického hlediska je *profil* Markovovým procesem (*další literatura např.: [19]*), který může být dán jako graf, jehož vrcholy představují jednotlivé *stavy* KA a jejich spojnice *přechody* těchto *stavů*, spuštěné odpovídajícími *událostmi*. Suma všech přechodů jednoho stavu je rovna 1, tedy 100%.



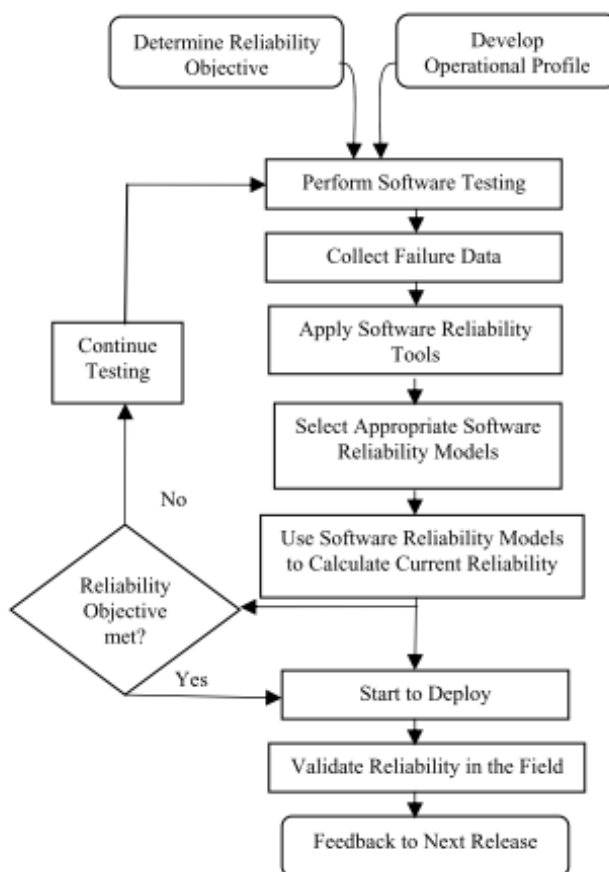
Obr. 16: příklad Markovova řetězce

[Zdroj: cnx.org/content/m34949/latest/MarkovChainCropped.png]

Teoretická spolehlivost testované implementace je dána vztahem $B = R^N$; ($N = \log_B(R)$). Tabulka 9 uvádí příklad počtu testů pro požadovanou spolehlivost testovaného produktu, při odhadované míře poruchovosti modelu. Klíčovou roli hraje proměnná B (horní mez pravděpodobnosti chybovosti návrhu). Jestliže neexistuje spolehlivý podklad pro její stanovení, jediný způsob, jak ověřit správnost takového odhadu je opakovaná iterace sady případových testů, případně její obměna (modifikace) mezi iteracemi.

Tabulka 9: Příklad vývoje případových testů

B horní mez pravděpodobnosti, chybovosti návrhu	R dolní mez odhadu spolehlivosti produktu	N počet náhodných případových testů produktu	Obecná míra spolehlivosti
0,007	0,999	5.000	střední spolehlivost
0,007	0,9999	50.000	
0,007	0,99999	500.000	



Obr. 17: Schéma procesu dokumentace a zvyšování spolehlivosti SW, zdroj: [16]

Další zdroje: např.: [16] shrnuje všeobecné informace na dané téma, mj. popisuje proces inženýringu spolehlivostních testů SW a [17] se zabývá reálnými výsledky modelování spolehlivosti a zvyšováním jeho kvality.

Z uvedených čísel plyne, že statistické testování spolehlivosti vyžaduje obrovské množství technických a časových zdrojů. Takový rozsah není možné realizovat manuálně. Jsou nutné nástroje pro automatické generování a následné spouštění testů. [1], kap. 5 uvádí platformu, která obsahuje GMT (Generic Modeling Toolkit) dostupné z www.isis.vanderbilt.edu/Projects/gme/.

Konfigurace se provádí, tvorbou aplikačně specifických metamodelů, které představují *operační profil* testovaného produktu a z něj vyplývající sadu testů. Modelovací jazyk popisuje především stavy KA a jejich konkrétní aplikací se modelují stavy testovaného SW. Následně se definují *přechody* těchto *stavů* (*třída událostí*, která *přechod stavů* spouští), očekávaná výstupní reakce – cílová hodnota *stavu* a očekávaná procentuální pravděpodobnost *přechodu stavu* v testovaném systému.

Testovací systém s procentuální pravděpodobností stejnou pro všechny kombinace v dané třídě, vygeneruje všechny kombinace *událostí*, vyplývající ze seznamu přípustných *přechodů stavů*. Pro vysoké počty přípustných kombinací *událostí* jedné *třídy*, disponuje nástroji náhodného výběru ze zadaného intervalu.

Dále testovací systém sestává z interpretru *operačních profilů* a generátoru testů. Interpret vygeneruje z *operačního modelu* sadu testů, které pro každou danou *třidu událostí* obsahují mj. výchozí stavy KA, počet případových testů, které mají být generovány (v závislosti na požadované spolehlivosti testovaného produktu – viz výše) a jednotlivé kroky případového testu. Generátor testů v souladu s touto specifikací vybere *třidu událostí* a z ní pak náhodně vybere *přípustné stavy* a vykoná testy.

Součástí výstupu generované sestavy případových testů je ukazatel hladiny významnosti, který reprezentuje pravděpodobnost, že definované stavy testovaného systému se budou střídát s náhodnou variabilitou. Obecně by úroveň tohoto ukazatele neměla poklesnout pod 20%, aby bylo možné testovací sadu prohlásit za kvalitní.

Přehled komerčních i volných testovacích nástrojů je dostupný např. zde:

www.cs.waikato.ac.nz/research/mbt/Tools.pdf

http://home.mit.bme.hu/~micskeiz/pages/modelbased_testing.html

4.4 TEST FRAMEWORK

Typicky knihovna programovacího jazyka, poskytuje procedury a funkce pro efektivní a rychlý vývoj případových testů. Využívají se k testování modulů a integračním testům.

Podle potřeby se framework nemusí držet formálních specifikací testů a jejich nároků. Může upřednostňovat důležitá, či poruchová místa implementace, její optimalizované chování, konkrétní specifikace zadavatele nebo poznatky a zkušenosti, které dosud nebyly promítnuty do formálních specifikací. Kromě formálních požadavků specifikací, mohou generovat rozšířené logy událostí k další optimalizaci a posílení implementace.

Testy mají být popsány: jednoznačné ID a jméno testu, zkoušené zařízení (funkce), očekávané chování, typické selhání, řízení toku (dat, událostí), zdroj.

Přehled testovacích framework, vč. základních vlastností a licenčního režimu:

http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks.

Přehled automatizovaných testovacích nástrojů: http://en.wikipedia.org/wiki/Test_automation.

II. PRAKTICKÁ ČÁST

1 PROPRIETÁRNÍ A VEŘEJNÉ KOMUNIKAČNÍ PROTOKOLY

Termín **proprietární** je ve světě SW obecně definován jako: „uzavřený“, jeho používání je podmíněno koupí nebo výslovným souhlasem vlastníka [9]. Ve vztahu k *protokolům* jde o stav, kdy je *protokol* vyvinutý a spravován konkrétním subjektem a jeho deklarace je:

- (a) utajena (dostupná pouze okruhu vlastníkem schválených osob);
- (b) má omezenou dostupnost a právo užití individuálním licenčním ujednáním.

1.1 Užití proprietárních komunikačních protokolů

Vlastník *proprietárního komunikačního protokolu* (PKP) jeho licenční, či faktickou nedostupností zajišťuje, že do systému s implementací daného PKP nelze připojit komponenty z produkce třetích stran, protože **nebudou schopny komunikace** s dalšími komponenty takového systému. Tím vlastník PKP sleduje:

- Ochranu obchodní zájmů (podpora prodeje vlastní produkce),
- Kvalitu a bezpečnost výrobků – ochranou stability a správné funkce výrobků a tím i ochranu dobrého jména svého a systému.
- Zajištění ochrany a odpovědnosti

1.1.1 KVALITA A BEZPEČNOST VÝROBKŮ

Znemožněním připojení zařízení třetích stran vlastník PKP omezuje chyby, které by mohly být způsobeny vzájemnou nekompatibilitou originálních zařízení a zařízení třetích stran.

1.1.2 ZAJIŠTĚNÍ OCHRANY A ODPOVĚDNOSTI

Vedle všeobecně rozšířené ochrany dat proti jejich zneužití neoprávněným přístupem, přečtením (využitím) a podvržením, jde v bezpečnostních informačních systémech (BIS) ještě o aspekt odpovědnosti. Výrobce zařízení musí podniknout k zajištění determinantní právní odpovědnosti opatření, která vyloučí pochybnosti v případě selhání takového zřízení nebo jeho částí a dále ve sporných případech, kdy by takové pochybnosti mohly nastat.

Vzhledem k tomu, že inženýring *protokolů* se vyvinul dříve a nezávisle na BIS, nejedná se o novinku. Podobná opatření, různými prostředky, v zásadě běžně uplatňují všichni výrobci informačních a telekomunikačních technologií (ICT).

Například lze uvést systém EPS; neautorizovaným připojením systémových komponent třetích stran (např. požárních hlásičů) by mohlo dojít k neprokazatelnosti příčiny selhání požární signalizace. Nebude jasné, zda příčina selhání byla na straně:

- výrobce originálního systému (vlastníkovi PKP) chybou konstrukce, či konkrétního kusu výrobku,
- výrobce nesystémového komponentu (třetí straně),
- provozovatele EPS, který nezajistil jeho řádný technický stav,
- dodavatelské organizace, která udělala chyby v instalaci/údržbě konkrétní aplikace.

Ačkoliv se na trhu vyskytují systémy EPS, do kterých je možné osadit požární hlásiče třetích stran, jedná se vždy o schválený a certifikovaný komplet – produktovou kooperaci založenou mezi výrobcí konkrétních dílů, poskytnutím komunikačního rozhraní – deklarace PKP – na základě licenčního ujednání.

Tato forma ochrany má za následek technicky nemožnou nebo obtížnou propojitelnost zařízení, který výrobce PKP a originálního dílu neschválil. Např. nelze podvrhnout laciný díl, pochybné výroby se zfalšovaným označením, jehož parametry neodpovídají originálnímu dílu a fakticky tím degradují celý systém nebo jeho část. I v případě, že by takový díl existoval (využitím metod reverzního inženýringu), jedná se v takovém případě o porušení autorského a obchodního práva ze strany dodavatele těchto dílů.

1.2 Vývoj proprietárních komunikačních protokolů

Každý *protokol* je v počátku svého životního cyklu proprietární. **Standardem** se stává po uvolnění jeho deklarace vlastníkem a přijetí odbornou veřejností, resp. výrobcí technologií. I za těchto podmínek může takový *protokol* stále (částečně) zůstat proprietární v tom smyslu, že jeho užití podléhá ochraně licenčních práv vlastníka.

Z hlediska teorie konstrukce *protokolů* při návrhu PKP nejde o nic nového. Konstrukční techniky popsané v teoretické části jsou platné obecně, tedy i pro návrh PKP.

1.2.1 PODMÍNKY VÝVOJE NOVÉHO KOMUNIKAČNÍHO PROTOKOLU

Rozhodnutí, zda vyvíjet PKP nebo jít cestou aplikace existujícího *protokolu* je v první řadě manažerský proces. Níže je rozhodovací model pro volbu řešení – vývoj nového PKP, úpravu stávajícího *protokolu*, aplikaci standardizovaného *protokolu*. Obecně lze říct, že toto rozhodování zahrnuje následující aspekty:

- Ekonomiku a pravděpodobnost úspěchu
- Projektový management
- Odbornou způsobilost a předpokládanou konstrukční kapacitu

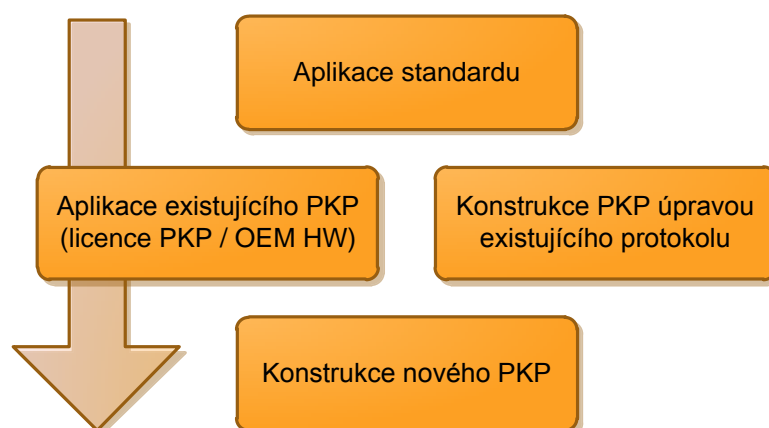
Kromě otázek odborné a projektově-manažerské způsobilosti, popsanych dále, je zřejmý ekonomický rozměr věci. Je třeba zohlednit i potenciálních škody, které mohou vyplynout z neodhalených konstrukčních chyb a jejich pravděpodobnost (kap. 4, resp. 4.3.2). Tu přímo ovlivňuje složitost konstruovaného PKP a zpětnovazebně tak roste objem výrobků, pro které je třeba PKP uplatnit.

Nový PKP má smysl konstruovat pouze, při splnění následujících podmínek:

- PKP bude uplatněn ve výrobku, jehož výrobní objem překročí kritické množství, definované jako celkový náklad na zajištění jeho dané funkcionality jinou cestou (aplikací obdobné, již existující, technologie třetí strany) v poměru k celkovým nákladům vývoje, odladění a certifikace nového PKP
- Ekonomická rizika neúspěchu (při vývoji, certifikaci a nasazení PKP) jsou, zvladatelná.
- Oportunitní náklady za vývoj jsou přijatelné s minimální akceptovatelnou mírou pravděpodobnosti úspěchu tohoto vývoje
- Potenciální hrozby škod plynoucích z rizika zbytkových vad nového PKP jsou pokryta, resp. zvladatelná nebo akceptovatelná.
- Neexistuje adekvátní technické řešení

Na Obr. 18 je schéma priorit výběru vhodné konstrukční varianty. Body 2 a 3 následujícího odstavce jsou na stejné úrovni.

1. Aplikace existujícího standardu
2. Aplikace existujícího PKP a zajištění souvisejících práv, přičemž cena nákupu licence by měla zohledňovat i cenu ne/zajištění zbytkových rizik plynoucích z konstrukce PKP jeho původcem, jak je uvedeno výše. Nezáleží, zda jde o formu licenčních práv nebo je právo užití PKP nakupováno jako součást dodávky OEM komunikačního zařízení.
3. Konstrukce vlastního PKP úpravou existujícího standardu nebo PKP. V případě nákupu licenčních práv k PKP by měly být přiměřeně aplikovány závěry předchozího bodu.
4. Konstrukce, vývoj a implementace nového *protokolu*, včetně testů a certifikace.



Obr. 18: Schéma priorit výběru konstrukční varianty

1.2.2 ROZHODNUTÍ O VARIANTĚ VÝVOJE

Je nutné podložit detailními analýzami, zahrnujícími nejméně následující závěry:

- (a) Definice a analýza technického zadání (**model požadavků a analýzy** – viz kap. 2.6.1 a 2.6.2). Je nezbytná k provedení rozhodnutí popsaných dále.
- (b) Analýza požadavků na budoucí **propojení výrobku (technologie) s okolím**. Pak je nutné řešit implementaci standardizovaných *protokolů*, nejlépe v souladu s referenčním modelem OSI (viz kap. 3 a 4). V současné době se standardem stává propojení využívající (obecně) rozhraní Ethernet dle IEEE 802.3, které umožňuje propojení s Internetem. Nespornou výhodou je jeho masové rozšíření a možnost geograficky nezávislé distribuce částí a komponent systémů kompatibilních s tímto standardem.
- (c) **Analýza standardizovaných protokolů a dostupných PKP** (které licenční podmínky dovolují použít) a rozbor, proč nelze žádný využít v projektovaném záměru nebo nepředstavuje vhodné, resp. ideální řešení (ekonomický a právní rozbor odchylek od ideálu).
- (d) Odborná **schopnost aplikovat** (případně) vybraný standardizovaný *protokol*, při míře jeho složitosti, která nemusí být dané aplikaci potřebná.
- (e) **Dostupnost potřebných kapacit** – personálu odpovídající kvalifikace a vývojových technologií; tj. schopnost danou věc zvládnout v určeném čase a odborné složitosti, při akceptovatelné míře rizika neúspěchu. Nároky jsou dány složitostí problematiky (počtu funkcí zamýšleného PKP a jejich variability, vrstevnatostí komunikačního modelu, složitostí konstrukční algebry atd.). Mírou

rizika není jen případná neschopnost daný projekt dokončit v plánované fázi, ale i rizika plynoucí ze zbytkových chyb (viz kap. 4.3.2). Zbytkové riziko rovněž představuje schopnost udržet životnost daného produktu (rozvoj, správa, podpora) po dobu nutnou k dosažení vytčených (ekonomických) cílů, tj. vytvoření odpovídajícího zisku, případně vlivu na trhu.

- (f) Je nutné zohlednit skutečnost, že konstrukce a aplikace PKP má negativní vlivy na **bezpečnost a spolehlivost výrobku**. Omezením dostupnosti PKP je jeho spolehlivost dána mírou hloubky provedených testů (viz kap. 4.) Zejména u složitějších *protokolů*, kde roste variabilita (*kompozitních*) *stavů* je tak omezena dostupnost nezávislého testování odbornou veřejností, resp. certifikačními autoritami; vývoj a implementace ověřených testovacích scénářů může být v některých případech v důsledku dražší, než vývoj samotného PKP.
- (g) Využitelnost **zařízení s implementací PKP na nižší komunikační vrstvě**, která umožňuje přenos uživatelsky definovaných dat. Jde zejména o *protokoly* výrobců (OEM) komunikačních modulů a technologií, produkováných k uplatnění v konstrukcích vyšší systémové úrovně třetích výrobců. Licenční právo jejich užití je svázáno s nákupem odpovídajícího HW výrobků.
- (h) Možnost **ochrany komunikace šifrováním**, při aplikaci standardizovaného *protokolu*. V praxi tato možnost může představovat zajímavý kompromis. Vzhledem k tomu, že většina komunikačních zařízení v současné době obsahuje mikroprocesor, lze šifrovací mechanismy efektivně aplikovat. Ty navíc mohou být dostupné v podobě framework, případně v rámci licencí GNU/GPL apod.

1.2.3 PRAKTICKÝ PŘÍSTUP K VÝVOJI PKP

K vývoji PKP má obecně smysl přistoupit, je-li výrobce v situaci, kdy neexistuje dostupné řešení (standardizovaný nebo proprietární *protokol*), kterým by pokryl své potřeby.

Vedle absence vhodného řešení jde o případy, kdy výrobce má k dispozici řešení založené na standardizovaném *protokolu*, ten však (některými) svými parametry nesplňuje potřeby dané aplikace. V tom případě je po všech stránkách (konstrukce, implementace, testování) efektivnější přistoupit k vývoji odvozeného řešení PKP, založeného na standardizovaném *protokolu* nebo jiném PKP, pokud to licenční podmínky umožňují. V praxi BIS jde o nejčastější postup.

Třetí modelová situace je, že výrobce využívá ke komunikaci OEM komponenty jiného dodavatele. Obecně se může jednat o zařízení na úrovni modulu nebo o čip, či sadu čipů, které již obsahují nějakou formu implementovaných standartních *protokolů*, nebo PKP a výrobce – spotřebitel OEM dílu provede pouze vývoj a implementaci koncového *protokolu* na vyšších *vrstvách*, např. na relační (5.) a/nebo aplikační (7.).

Rozhodnutí o vývoji nového, či odvozeného *protokolu* ovlivňují zejména dále uvedené faktory:

- **Výkon:** existuje-li reálný předpoklad, že produkt vývoje bude výkonnější, než dostupná řešení, resp. rozsah odvozeného vývoje omezit na nezbytné minimum za respektování tohoto hlediska. Takovým případem může být i zjednodušení existujícího standardu o některé funkce a datové struktury.
- **Dostupnost funkcí:** některé z potřebných funkcí nejsou dostupné v existujícím *protokolu*, který jinak vyhovuje zamýšlenému účelu. Pak je možné vytvořit – v souladu s OSI modelem – *podvrstvu* v tomto *protokolu* (viz kap. 3.4.3) nebo jinou funkční úpravu; v případě více funkcí, které vytváří logický, zapouzdřený, celek (*třídu*) je logické vyvinout nový, doplňkový, *protokol*.
- **Aplikace v přenosovém médiu:** v tomto případě jde o nový *protokol* na 1. (fyzické) resp. 2. (linkové) vrstvě. Tento případ je dnes spíše otázkou aplikovaného výzkumu, nicméně v praxi BIS byl ve své době velmi častý a řada *protokolů*, které dnes vytváří standard komunikace v BIS, pracovala (a pracuje) právě na těchto dvou vrstvách, jak patrné z PŘÍLOHA 4. Důvody k tomuto přístupu byly komplexní; od zřetele na ekonomickou nenáročnost takového řešení k jeho dostatečnému výkonu vzhledem k řešené složitosti.
- Shora popsaná **ekonomická hlediska a rizika**
- Při volbě vyvíjet nový PKP je třeba mít na paměti rovněž skutečnost, že pokud má tento PKP být součástí komplexního systému (např. OSI), bude se od něj pro dosažení plné kompatibility vyžadovat kooperace s řadou dalších – již standardizovaných – protokolů, které pracují na stejné nebo jiných vrstvách OSI modelu a plní jiné funkce důležité pro komplexní činnost systému, např. reakce na hlášení poruch, QoS apod.

Zřejmě nejmarkantnějším příkladem takového přístupu je telemetrická signalizace prováděná cestou Internetu, kdy je plně využíván zásobník TCP/IP protokolů a na relační

vrstvě je proveden vývoj PKP, pro jehož přenos je následně vyžito kompletních služeb TCP/IP. Příkladem může být přenos protokolů CID a SIA z prostředí telefonních okruhů – viz PŘÍLOHA 4.

2 POŽADAVKY NOREM NA KOMUNIKACI BIS

Tato kapitola popisuje požadavky kladené na komunikaci a signalizaci BIS z hlediska platných norem v ČR. Sumarizuje jednotlivé systémy a normy, které definují nějaké požadavky na komunikaci BIS z hlediska *protokolů*.

2.1 I&HAS

2.1.1 KOMUNIKACE I&HAS DLE ČSN EN 50131-1

Norma definuje základní požadavky na systémy I&HAS, jako funkční celek, jeho komponenty a součásti. I&HAS obecně představuje programovatelný *stavový stroj* řízený událostmi, které nastávají působením vnějšího vlivu na senzory detektorů, nebo jako výsledek mechanismu sledujícího výskyt, definovaného vnitřního stavu systému. Ohledně *komunikace* těchto systémů je kladen důraz na včasnost (dostupnost), integritu a důvěryhodnost předávané informace.

Definuje povinnost *komunikace* určených *stavů* a *událostí* komponent systému, jestliže jsou splněny iniciační (*ochranné*) podmínky a požadavky na fyzické propojení:

- Minimalizovat rizika zpoždění, modifikace, záměny a ztráty zpráv. I&HAS bezpečnostní třídy 4 musí mít prostředky detekce těchto rizik. Riziky norma souhrnně definuje bezpečnost komunikace (kap. 8.8.5).
- Schopnost komunikace (na žádost) musí být ověřitelná *monitorováním propojení periodickou komunikací* a musí být možné vynutit komunikaci na sdílené systémové sběrnici tak, aby *signalizace* proběhla v očekávaném čase a kvalitě.

Zdroj: [20].

2.1.2 KOMUNIKACE DETEKTORŮ I&HAS DLE ČSN EN 50131-2-x

Definuje požadavky na komunikaci detektorů I&HAS, které primárně zjišťují a předávají informaci o *poplachu* – nebezpečí pro život, majetek nebo okolní prostředí ([20] 3.1.5).

Níže jsou popsány požadavky na komunikaci dle ČSN EN 501313-2-2 [21]. Ostatní části souboru ČSN EN 50131-2-x, popisující další typy detektorů jsou obdobné.

[21] připouští **signalizaci elektrickým signálem** nebo **zasíláním zpráv**; zprávy se nesmějí překrývat (docházelo by k rušení).

Komunikovat mohou i další systémové komponenty, zejména výstražná zařízení (ČSN EN 50131-4) a napájecí zdroje (ČSN EN 50131-6), od kterých se, mimo primárních funkcí, v určitých případech vyžaduje zpětné *hlášení* jejich *stavů*.

Ustanovení [21] kap. 4.1 pokrývá **signalizaci konvenčních** detektorů, vybavených logickým rozhraním v podobě relé s NC nebo NO kontakty. Detektory komunikují změnou hodnoty (typicky) činného odporu, která nastane v bezpotenciálovém *propojení* detektoru a řídicí ústředny při (*události*) sepnutí kontaktu určeného relé. Událost je vyvolaná vnitřním *stavem* detektoru (naměřenou úrovní sledované veličiny nebo výskytem sledovaného jevu). Kombinací zapojení rezistorů v *propojení*, které se realizuje na rozhraní detektoru, je možné signalizovat 2, 4, resp. 8 zpráv, při zapojení jednoduše nebo dvojité vyvážené smyčky, resp. smyčkou ATZ.

Základní výhodou této koncepce je jednoduchá konstrukční implementace, univerzální rozhraní detektorů různých výrobců a tím jejich zaměnitelnost. To je vyváжено množstvím nevýhod jako omezení počtu signalizovaných *stavů* (detektoru), nároky na kabelové vedení, nemožnost chránit přenášené *zprávy* proti podvržení, omezenou vzdáleností od ústředny systému, resp. expandéru (distribuované moduly smyčkových vstupů). Vůči zapojení některé ze sériových sběrnic je tato instalace i složitější.

Stejná kapitola normy pokrývá **komunikaci zasíláním zpráv** systémovými sběrnicemi. Ty jsou dnes typicky založené na sériové komunikaci. Podle výrobce se využívá různých *protokolů*, příklady uvádí PŘÍLOHA 4. V minulých generacích I&HAS se tato technologie uplatňovala pro složité komponenty (klávesnice, expandéry), kde byla potřeba instalace ve velké vzdálenosti od ústředny a větší objemy komunikovaných *hlášení*. V posledních letech výrobci přistoupili k vývoji **sběrniceových detektorů** – vybavených rozhraním pro komunikaci sběrnicí.

Všechny takové komponenty využívají PKP a v tomto smyslu se na ně vztahují závěry kap. 1, tzn. komunikaci lze zabezpečit proti podvrhu i zneužití, komponenty systémů nejsou volně zaměnitelné, mají významně vyšší funkcionalitu.

Většina současných systémů představuje kombinaci obou přístupů. Umožňuje na vstupech aplikovat konvenční detektory s eklektickou signalizací, tak systémové detektory, připojené na systémové sběrnici.

2.1.3 BEZDRÁTOVÁ KOMUNIKACE DLE ČSN EN 50131-5-3

Tato část normy řeší požadavky na bezdrátovou komunikaci, komponentů I&HAS umístěných v chráněném prostoru a na krátké vzdálenosti. Lze je obecně rozdělit do dvou kategorií – požadavky na rádiovou signalizaci a požadavky na přenosový *protokol*.

Požadavky na rádiovou signalizaci:

- Možnost omezení vysílaného výkonu při definovaných podmínkách
- Omezení přístupu ke kanálu v určeném časovém rámci
- Povinnost obousměrné komunikace komponent systémů 3. A 4. Bezpečnostní třídy
- Max. chybovost v objemu doručených zpráv
- Omezení rizika kolize zpráv
- Zajištění propustnosti omezením počtu chybně doručených zpráv
- Minimální odolnosti proti rádiovému rušení a jeho detekci
- Trvalé monitorování rušení rádiových kanálu a signalizace rušení
- Periodickou komunikaci k ověření funkčnosti propojení

Požadavky na přenosový protokol:

- ID každého komunikačního prvku,
- Rozpoznat záměnu prvku pro systémy 3. A 4. Bezpečnostní třídy
- Šifrování komunikovaných zpráv a minimální odolnost šifry proti prolomení (podvržení zprávy)
- Ověřování doručených zpráv pro systémy 3. A 4. Bezpečnostní třídy
- Signalizace ztráty spojení odmlčením komponentu

Zdroj: [22].

2.2 Systémy kontroly vstupu (SKV)

ČSN EN 50133-1 [23] definuje minimální sadu událostí, které musí být systémem přenášeny a indikovány a dovolené zpoždění přenosu informace. Dále chování systémů při přerušení komunikačních cest. Další požadavky na komunikaci a *protokoly* nedefinuje.

2.3 Systémy průmyslové televize (CCTV)

2.3.1 ČSN EN 50132-1

Definuje obecné požadavky na přenos signálů nesoucích informace zachycené kamerou. Připouští digitální a analogový přenos a uvádí obvyklá přenosová média (kroucenou dvojlinku, koaxiální kabel, optické vlákno). Přímou připouští možnost bezdrátového přenosu a sdílené využití IP (paketových) sítě – tato technika je rozpracována v dalších částech normy.

Definuje požadavky na integritu všech dat, tedy i přenášených. Od kontroly funkčnosti přenosových tras, přes požadavky na identifikaci přenášených DU, důvěryhodná časová razítka, prevenci podvrhu a zamezení neautorizovaného přístupu k systému.

Toky dat se nesmí ovlivňovat a musí být minimalizováno jejich zpoždění. Pro systémy kategorie 4, které využívají sdílená propojení, musí být možnost konfigurace max. šířky přenosového pásma pro přenos obrazu ze zdroje (kamery) a pro přijímací, resp. zobrazovací technologii.

Zdroj: [24].

2.3.2 ČSN EN 50132-5

Popisuje přenosový systém (kanál) z hlediska parametrů přenosu analogového video signálu (vstupní a výstupní impedance, povolený útlum, úroveň vstupního a výstupního signálu, úroveň stejnosměrné složky, kmitočtovou charakteristiku přenosového média, dovolené zkreslení, poměr signál šum, zisk, časové zpoždění a další). Definuje požadavek dostupnosti propojení na úrovni 99,999%. *Zdroj: [25].*

2.3.3 ČSN EN 50132-5-1

První část definuje požadavky na konektivitu pro přenos videa využitím IP sítě. Druhá část definuje výkonové požadavky komunikace – časování, kvalitu a dostupnost. Velkou pozornost věnuje QoS, zejména v prostředí sdílené sítě. Pod QoS zahrnuje především: zpoždění, kolísání zpoždění, šířku přenosového pásma a dostupnost na úrovni 99,999%, (stejnou jako u analogového přenosu) a uvádí i techniky k dosažení uvedené dostupnosti (např. redundance, load balancing a sdílení zdrojů). Nakonec jmenuje možné varianty video streamingu (např. RTP over UDP/IP, http over TCP/IP) a podporu pro management sítě pomocí SNMP. Striktní požadavky klade na bezpečnost přenosu (šifrování,

autentifikace, kontrola integrity apod.) a na dostupnost časové synchronizace přenosových (a přijímacích) zařízení, při využití (v prioritách po sobě): NTP protokolu (RFC1305), SNTP protokolu (RFC2030), resp. časového serveru (RFC868).

Předpokládá oddělené datové toky pro různé účely přenosu (obraz, zvuk, řídicí a kontrolní data) a pro různé situace (prioritní provoz ze zasažených míst). *Zdroj: [26].*

2.3.4 ČSN EN 50132-5-2

Definuje *zásobník protokolů* a služeb pro CCTV, založené na IP protokolu a existujících standardech ve vyšších *vrstvách*. Je přímo určena pro systémy založené na sdílených paketových sítích. PŘÍLOHA 4 uvádí některé *protokoly*, které přebírá a jejich význam v BIS obecně. Na těchto standardech staví *IP Video Standard Protocol* pro přenos videa a souvisejících informací paketovými sítěmi. Souvisejícími informacemi jsou např.: časová synchronizace, obecné řídicí instrukce, instrukce PTZ (řízení polohy a zoomu kamer), zvuková data atd.

Norma je výsledkem spolupráce organizací ONVIF a PSIA, které představují průmyslová sdružení s cílem definice a prosazení interoperabilního standardu CCTV technologií založených na IP *protokolech*.

Základem je definice interoperability CCTV systémů (schopnosti integrovat zařízení různých výrobců). První část popisuje *protokoly* a jejich význam pro přenos obrazových a souvisejících dat. Druhá popisuje *protokoly* podporované ONVIF, třetí *protokoly* podporované PSIA. Uvádí rovněž *protokoly* pro QoS, Device Discovery, video kodeky a transport videa k dosažení kooperace IP CCTV.

2.4 Systémy elektrické požární signalizace (EPS)

Obecně se norma EN 54 nezaobírá specifikací konkrétních podmínek komunikace a z toho plynoucích nároků na *protokoly*. Namísto toho popisuje stavy, do kterých se systém nebo jeho komponenty smí, či nesmí dostat (jak je uvedeno výše) a jak na tyto stavy systém musí reagovat.

2.4.1 ČSN EN54-2

Popisuje ústřednu EPS a z hlediska požadavků na vlastnosti přenosových cest, povinnost přijmout a vyhodnotit signály z požárních úseků. Signály se nesmí ovlivňovat a mají definováno max. dovolené zpoždění. Požaduje se možnost sériové detekce (poplach je

vyhodnocen na základě současné detekce více detektory) Definuje množinu přenášených signálů v požárních (detekčních) smyčkách a skrze normalizované I/O rozhraní: {požár, porucha, vypnuto, signalizace ze všech sekcí, zrušení poplachu}. (Zdroj: [30]).

2.4.2 ČSN EN 54-13

Definuje, že porucha na přenosové cestě mezi ústřednami nesmí ovlivnit funkci ústředny a komponenty, musí pracovat v krajních mezích. Porucha jedné přenosové cesty nesmí ovlivnit ostatní. Max. interval ztráty spojení přenosových cest je povolena na 300s a za tuto dobu musí dojít k zotavení (nejméně na úrovni úseku odděleného izolátory). Dále je požadovaná jasná identifikace přijímaných signálů. (Zdroj: [28])

2.4.3 ČSN EN54-21:

Popisuje poplachové přenosové zařízení. Odkazuje na soubor norem EN 50136, které popisují komunikaci monitorovaných objektů a DPPC (viz kap. 2.6).

Dále definuje max. přenosový čas a EMC parametry zařízení. (Zdroj: [31])

2.4.4 ČSN EN 54-25

Popisuje podmínky rádiové komunikace komponent EPS. Definuje odolnost proti náhodnému útlumu a *protokol* (obecně) musí zajistit odolnost zprávy proti ztrátě. Přenášené zprávy požaduje opatřit jedinečnou identifikací (adresou) komponentu a přijímač musí být odolný proti záměně komponentů mezi různými systémy, pracujícími ve vzájemném dosahu. Požaduje detekci ztráty spojení komponentu a ústředny. (Zdroj: [29])

2.5 Systémy přivolání pomoci

2.5.1 ČSN EN 50134-1

Vyjmenovává bloky systému přivolání pomoci – z hlediska komunikace a *protokolů* jsou podstatné: poplachový přenosový systém a DPPC, (viz kap. 2.6). Dále musí být systém vybaven prostředky obousměrné hlasové komunikace. (Zdroj: [32])

2.5.2 ČSN EN 50134-2 A 3

Definuje požadavky na *aktivační zařízení* (tísňový hlásič) a *kontrolér* (lokální přijímací zařízení). Z hlediska komunikace připouští bez/drátový přenos. V bezdrátovém provedení

musí mít kombinační kód délky nejméně 1 z 256 možností. Bez rozdílu druhu komunikace musí přenášet informaci o poruše a mít jednoznačný identifikátor zdroje zprávy.

Bezdrátové provedení má pracovat s výkonem 0,1 až 2 mW.

Každý komponent předává informace nadřazenému komponentu – aktivační zařízení systémovému kontroléru a ten DPPC. Oba standardy specifikují povinné typy informací a podmínky spolehlivého přenosu zprávy (dovolené časové zpoždění a případné zálohování přenosové trasy). (Zdroj: [32], [34])

2.5.3 ČSN EN 50134-5

Přímo odkazuje na standardy komunikace DPPC (soubor ČSN EN 50136 – viz kap. 2.6). Z hlediska komunikace připouští libovolnou přenosovou síť a libovolné přenosové médium; požaduje splnění požadavků kategorie A3 pro komunikaci s DPPC dle [36] (dostupnost 99,5%) a dále podle této normy definuje požadavky přenosových časů a času detekce poruchy. Specifikuje frekvenční charakteristiku přenosového kanálu, pro přenos hlasové komunikace. (Zdroj: [35])

2.6 Systémy dohledového a přijímacího poplachového centra (MARC)

Klíčovou normou z hlediska komunikace a *protokolů* nyní představuje nově vydaná ČSN EN 50136-1, která nahrazuje všechny předchozí části souboru ČSN EN 50136-1-x. Přímo odkazuje na výše popsané ČSN EN 50131-1 [20], EN 54-21 [31] a ČSN EN 50134-1 [32]. Zahrnuje požadavky na propojení a jeho výkonnost mezi BIS instalovaným na straně střeženého zájmu (objektu, osoby) a DPPC.

Z hlediska odkazů na další normy stojí za povšimnutí soubor ČSN ISO/IEC 10118 – Bezpečnostní techniky – HASH funkce, které využívá k zabezpečení, resp. podpisu přenášených zpráv.

Z hlediska přenosového média a typu sítě norma nerozlišuje. Explicitně uvádí, že poplachový systém – kterým vnímá jakékoliv zařízení přenášející informaci o poplachu nebo informaci s tímto jevem související – musí být schopen využívat přenosové soustavy bez rozdílu druhu – za předpokladu existence kompatibilního rozhraní, a že tyto přenosové cesty mohou být (obecně) sdíleny a z toho titulu jsou nespolehlivé a zatíženy poruchami. Provoz údržba a poruchy těchto přenosových tras nesmí bránit naplnění účelu normy.

Dále definuje možnost využití alternativních přenosových tras, přičemž u každé předpokládá telekomunikační rozhraní pro účely využití v poplachovém přenosovém zařízení. Podle tohoto kritéria následně hodnotí kategorie poplachového přenosového systému, přičemž u 4 z nich přímo předpokládá využití alternativní přenosové cesty. Pokud je alternativní přenosová cesta využívána, musí existovat i alternativní přijímač a jeho síťové rozhraní na straně DPPC.

Zavádí pojem DoS (Denial of Service – viz např.: http://en.wikipedia.org/wiki/Denial-of-service_attack), který pochází z IP sítí a prostředí Internetu, ale obecně se dotýká jakékoliv veřejně telekomunikační sítě. Od koncových přijímacích zařízení vyžaduje schopnost detekce takového útoku a generování poplachové zprávy při jeho odhalení. Říká, že žádná jiná zpráva nebo služba nesmí ovlivnit přenos poplachové zprávy, čímž mj. předpokládá prioritizaci provozu.

V žádném dalším ohledu norma neřeší nároky na *protokoly*. Místo nich stanovuje sérii parametrů:

- doba přenosu, její průměr na úrovni 95 percentil a povolené maximum
- interval hlášení
- monitorování spojení
- dostupnost služby přenosu poplachu, která se v měřených kategoriích pohybuje od 97 do 99,9%
- požadavek na potvrzování doručení přenášených zpráv
- povinnost šifrování přenášených zpráv na úrovni 128 b délky symetrického klíče; dále zde nařizuje expertní přezkoumání používaného algoritmu, pokud se nepoužívá standard (např. AES)
- zabezpečení proti substituci, neautorizované manipulaci a modifikaci přenášených dat, jako lehčí formu bezpečnosti (oproti šifrování) u zpráv, kde data nemají kritickou povahu (např. využitím HASH).

2.7 Shrnutí

Ze shora uvedených částí kapitoly 2 plyne, že platné oborové standardy neřeší problematiku *komunikačních protokolů* BIS přímo. Místo toho nechávají volný prostor alternativní konstrukci a definují a popisují stavy, do kterých se systémy a jejich

komponenty smí, či nesmí dostat. Dále definují základní požadavky na komunikaci těchto systémů z hlediska zajištění sledovaného standardu bezpečnosti, resp. zabezpečení.

Z legislativy je vidět, že standardy novějšího vydání jednoznačně kopírují trendy výroby, které jdou cestou sbližování BIS a ICT. Pokračování tohoto trendu v následujících letech lze očekávat. Z těchto norem je rovněž patrné, že nové technologie BIS čerpají z již zavedených a osvědčených standardů na úrovni protokolů i na úrovni standardů zabezpečení přenášených dat. Tato skutečnost je logická, protože většina současných, standardizovaných, protokolů byla konstruována abstraktně, čili z hlediska funkcionality, nikoliv s ohledem na konkrétní povahu přenášených dat.

Naopak starší standardy vypovídají o praxi své doby, která využívala více analogových technologií a soustředila se na lokálně propojené systémy, resp. jejich komponenty. Lze tedy konstatovat, že tyto standardy pracují až na výjimky s 1. A 2. Vrstvou modelu OSI, když definují požadavky na fyzickou vrstvu a linkové propojení, které přepokládají v některé s běžně používaných architektur (sběrnice, kruh, hvězda). Totéž lze tvrdit i o požadavcích na komunikaci rádiových technologií.

V obou případech popsaných shora, tj. technologiích přebírajících existující standardy paketových sítí i v uzavřených architekturách pracujících v lokálních *propojeních* lze úspěšně využít techniky modelování *protokolů* UML jazykem, popsané v teroristické části. Obecné požadavky na komunikaci uzavřeného *propojení* totiž představují jednotlivé stavy KA a *stavových strojů*, ke kterým standardy definují *ochranné podmínky*, resp. dovolené a zakázané *stavy*.

Např. celistvost komunikační linky EPS systému lze popsat jako *kompozitní stav* vzniklý na základě výsledku *stavu* ověření její celistvosti a obdobně *stav* přerušení této linky je *kompozitní stav* obsahující *stavy* nedostupnosti určitých funkcí a *stavu* časovače ústředny, který nesmí přesáhnout dovolených 300 s. Analogicky *stav* zahlcení rádiové trasy bezdrátových komponent poruchovými zprávami obsahuje *kompozitní stav* složený ze *stavů* nedoučení očekávaných zpráv v předepsaném čase a příjmu určitého objemu nekonzistentních dat. Z nich se pak v *modelu* odvíjejí další *toky* a *stavy*, které dosahují požadavků norem eliminací těchto zakázaných stavů.

3 VRSTEVNATOST V PRAXI BIS

Jak popisuje kap. 3 teoretické části, techniky vrstvení protokolů slouží k jejich kombinaci v rámci zajištění skupin (*tříd*) určitých funkcí *protokolů* a skrze něž kooperují entity na aplikační – nejvyšší – vrstvě. Jak vyplývá z kap. 4, výrobci BIS využívají mechanismů vrstevnatosti *protokolů* na dvou úrovních:

1. K dosažení otevřenosti svých systémů a tím jejich propojení s okolním světem
2. K optimalizaci PKP pracujících na vnitřní úrovni těchto systémů

Model OSI obecně nabývá v BIS významu tak, jak se soudobé technologie stále více sbližují s ICT a otevírají se sdíleným komunikačním prostředkům. Nové standardy v BIS (viz např. [36] a [27]) jednoznačně předpokládají využití IP sítí a prostředí Internetu, což se neobejde bez detailní znalosti této problematiky.

Vrstevnatost dále pomáhá s adaptací historicky zavedených protokolů, jakou např. Contact ID a SIA (viz PŘÍLOHA 4), využívaných při komunikaci BIS s DPPC. Jejich logická (uživatelská) datová struktura je na 5. Vrstvě OSI, přenesena do prostředí Internetu, resp. IP sítí. Podle způsobu implementace buď zůstává na 5. Vrstvě nebo se přesouvá na 6. (např. v případě použití šifrování, resp. specifických způsobů kódování), ale je nezávislá na způsobu zajištění přenosu nižšími vrstvami, které tyto služby poskytují; výrobce se touto problematikou nemusí zabývat – poskytují ji sdílené prostředky ať na úrovni služeb nebo na úrovni komunikačních zařízení třetích stran (OEM), (např. zařízení na protokolu ZigBee, viz PŘÍLOHA 4). To je smyslem OSI modelu a základním předpokladem kompatibility technologií, jak z hlediska rozličných výrobců, tak z hlediska právě popsané přenositelnosti **skupiny** určitých logických funkcí.

Z jiného úhlu pohledu – vývojář zajistí propojení aplikační výměny (tedy na 5, vrstvě OSI) a již neřeší, zda bude přenos realizován Internetem,

Techniky vrstvení mj. umožňují řešit diversifikaci výroby a ochranu autorských práv, včetně aplikace PKP. Typickým příkladem může být výrobce ZigBee modulů standardu IEEE 802.15.4, který poskytuje komunikační platformu a vývojář vyšší úrovně aplikuje tyto moduly spolu se standardy na odpovídajících vrstvách modelu OSI + vlastní PKP např. na 5. Vrstvě.

4 PRAKTIKY KONSTRUKCE PKP VÝROBCŮ BIS:

Tato kapitola popisuje konkrétní přístup výrobce BIS ke konstrukci PKP a využívání standardizovaných *protokolů*. Jedná se o proprietární zásobník *protokolů* výrobce JABLOTRON ALARMS, a.s. (dále JA) Zdrojem informací jsou osobní konzultace s pracovníky vývojového oddělení této společnosti a s jejich laskavým svolením jsou zde použity.

4.1 Základní koncepce

Všechny vytčené cíle se snaží realizovat při minimalizaci výrobních, tudíž i vývojových nákladů a v obecné rovině striktně sleduje poměr cena/výkon, při dosažení požadovaných technických parametrů.

Rozeznává dva přístupy:

- **Vnitřní prostředí systému:** je proprietární a soustředí se na zvládnutí všech konstrukčních aspektů, vč. aplikace s konkrétní (specifickou) součástkovou základnou. Sleduje systémová hlediska – především bezpečnost a stabilitu – a hlediska obchodně ekonomická (požadavky na funkcionalitu jednotlivých komponentů, systému jako celku a jejich užití – např. aspekty snadné a bezproblémové instalace).
- **Rozhraní s vnějším světem:** které musí zachovávat otevřenost k návazným systémům třetích stran; v tomto smyslu sleduje architekturu OSI modelu a kompatibilních *protokolů*.

4.1.1 DŮVODY PRO VÝVOJ PKP:

Celé vnitřní prostředí stojí na PKP vlastní konstrukce. Klíčové důvody jsou následující:

- optimalizace vlastních funkcí
- dosažení vysoké energetické účinnosti celého systému (vysoká datová režie standardizovaných *protokolů* snižuje energetickou účinnost, vysoké přenosové rychlosti snižují účinný dosah komunikace při dané úrovni energetické spotřeby)
- 100% podpora kmenových výrobků společnosti
- Ve vyráběném objemu celkově vysoké náklady za licenční poplatky standardizovaných *protokolů*

4.1.2 DŮVODY PRO VOLBU STANDARDIZOVANÝCH PROTOKOLŮ

Kromě dosažení požadované otevřenosti a propojitelnosti na úrovni *vnějšího rozhraní* jsou to důvody ekonomické:

- Tam kde složitost protokolového zásobníku přesahuje úroveň, kterou je únosné vyvíjet a implementovat vlastními prostředky – vzhledem k objemu výroby a dostupnému času
- Zjednodušení vývoje nových systémů – ověření základní funkcionality, průzkum trhu, vyjasnění zadání; malosériová výroba; tj. ve fázích kde nemá smysl nebo není prostor optimalizovat komunikace.

4.2 Techniky modelování a vývoje

Modelování je založené na teorii KA a využívá vybrané části UML, které poskytují potřebné přístupy a objekty. Modely se provádějí po částech, jak je třeba je zpracovávat s ohledem na průběh vývoje.

Techniky modelování sledují bottom-up přístup (odzdola-nahoru), což nejlépe odpovídá sledovaným cílům, které ve struktuře výrobků společnosti představují:

- Jasná determinace proveditelnosti HW a SW konstrukce
- Cenová a energetická omezení pro volbu součástek (vliv na cenu výroku a limit celkové energetické náročnosti každého komponentu)
- Konečný výpočetní výkon procesorů přípustných do konstrukce podle přechozích omezení a jejich další technická omezení (dostupnost a spolehlivost konkrétních vnitřních funkcí)
- Odolnost *protokolů* proti rušení (vlastní praktické zkušenosti a požadavky norem – viz [22])
- Minimalizace datových režii s ohledem na limitní propustnost pásma a energetickou účinnost
- Budoucí zaměnitelnost součástek (nahrazení novou generací)
- Budoucí rozšiřitelnost modelu (vhodná počáteční volba kapacity *protokolu*)
- Max. logická jednoduchost modelu *protokolu* ve vyšších *vrstvách*

V případě, že jsou dostupná otevřená řešení, na kterých lze vývoj založit, vychází z těchto řešení a vytváří odvozený PKP.

4.2.1 MODEL PROTOKOLŮ

Obecně se jedná o čtyř-vrstvý model (viz kap. 0) s jedním logickým *protokolem* a odpovídajícím protokolem na *fyzické vrstvě*. Typ protokolu determinuje typ (struktura) jeho hlavičky.

Délka DU *protokolu* vychází z technických omezení *fyzické vrstvy* na úrovni optimální volby mezi aspekty: modulace, přenosové rychlosti a spolehlivosti.

Spíš než vytváření nových *protokolů*, (resp. *vrstev*) JA využívá podvrstvy a verzování *protokolů*, které společně poskytují potřebnou flexibilitu. Funkce *vrstev* jsou v logickém *protokolu* zařazeny sériově za sebe, což přináší optimalizaci z hlediska datové režie, tudíž i energetických nároků komunikace.

Jednotlivé verze *protokolu* jsou následně nasazovány podle konkrétních aplikačních potřeb (funkcí konkrétního výrobku).

4.2.2 ZÁSObNÍK PROTOKOLŮ:

JA pro své výrobky využívá model protokolů o čtyřech vrstvách. Vychází ze základních principů OSI a podle potřeby implementuje funkce jednotlivých *vrstev* OSI. Jejich zapouzdření ve *vrstvách* tohoto zásobníku volí s ohledem na optimalizaci (podle nejlepšího provozního výkonu a celkové energetické efektivity, která hraje v bezdrátových technologiích významnou roli). Z těchto kritérií vychází i filozofie konkrétních *protokolů*, jak je uvedena v kap. 4.2.1, protože klasické vrstvení *protokolů*, jak jej uvádí model OSI, vyžaduje separátní hlavičku pro každý *protokol*. To se však v podmínkách omezené přenosové kapacity pásma a vysokých nároků na energetickou účinnost komunikace jeví jako nevýhodné.

Hranice *vrstev* (*třídy* funkcí) jsou určeny potřebnou funkcionalitou v dané *vrstvě* a způsobem kódování DU.

Předpoklad zařazení dalších vrstev výrobce považuje spíše za teoretický. Místo toho využívá dělení skupin funkcí *protokolu* do *podvrstev*. Nová *podvrstva* (resp. nový *protokol*) jsou zařazeny v okamžiku, kdy stávající prostředky nedokáží požadované funkce plnit vůbec nebo optimálně s ohledem na cíle technik modelování – viz kap. 4.2.

Tabulka 10: Model zásobníku protokolů JA

Vrstva	Název		Význam
1.	Fyzická	↓	Zjišťuje základní komunikaci po zvoleném přenosovém médiu
2.	Linková	↑	Zajišťuje abstrakci fyzické vrstvy pro vyšší funkce.
			Zapouzdřuje a obsluhuje různá média (drátové a bezdrátové sběrnice), propojuje různé typy HW
		↓	Na rozhraní k vyšší vrstvě zajišťuje šifrování dat. Důvody pro umístění této funkcionality do této úrovně leží v optimalizaci výkonu a algoritmizace.
3.	Komunikační		Z hlediska OSI představuje síťovou a hybridní transportní Zajišťuje funkce doručení dat a session
4.	Aplikační	↑	Podle druhu dat (obraz, zvuk, telemetrická a systémová data) zajišťuje přípravu jejich směrování (výběr cíle), případně užitý protokol.
		↓	Zpracovává a připravuje data produkovaná detektory systému, zajišťuje obsluhu paměti.

Model na první pohled připomíná zásobník TCP/IP, běžně používaný pro Internet. Architektura však nasazuje různé funkce na různé vrstvy tak, aby odpovídaly potřebám výrobce, případně – jak je uvedeno výše, některé funkce slučuje a minimalizuje počet aplikovaných *protokolů*.

5 KOMUNIKAČNÍ PROTOKOLY V PRAXI BIS

Tato kapitola uvádí přehled důležitých standardizovaných protokolů a jejich význam v praxi BIS. Přehled uvádí, jak protokoly používané v internetovém prostředí, tak protokoly ze světa sítí spojovaných okruhů a specifické aplikační protokoly BIS.

Přehled je předmětem PŘÍLOHA 4.

ZÁVĚR

Bakalářská práce přináší teoretický rozbor teorie a praxe komunikačních protokolů v BIS, jejich konstrukce a aplikace. Zabývá se metodikou konstrukce *protokolů* využitím UML jazyka, který patří mezi uznávané standardy a dále představuje nástroj k modelování stavových procesů a chování systémů (soustav procesů) obecně. Proto se jeho využitelnost neomezuje jen na konstrukci protokolů, ale může přesahovat do konstrukce SW a procesů obecně. Oba tyto obory jsou v BIS neméně potřebné a neustále i v tomto oboru nabývají na významu.

Spojuje se zde obecná teorie UML vysvětlená např. v [4] a aplikací UML pro konstrukci *protokolů* (viz [1]), aplikovaná konkrétním modelovacím nástrojem (StarUML) dostupným pod licencí GNU/GPL [6], aby bylo dosaženo nezávislosti na nástrojích komerčních, které mohou přinášet i určité specifické odlišnosti.

Dále teoretická část vysvětluje techniky modelování a významu vrstevnatosti *protokolů*. Toto téma nabývá na významu, jak se BIS sbližují s ICT a přebírají technologie otevřených systémů, propojitelných Internetem. Kromě toho poskytuje důležitý pohled na vnímání uzavřených (systémových) komunikací BIS.

Naposledy jsou uvedeny jak konvenční, tak trendové metody testování *protokolů* tak, aby si čtenář vytvořil přehled o významu tohoto oboru ve smyslu možností ověřování spolehlivosti *protokolů* ne jen při vývoji technologií, ale i při jejich aplikaci a přebírání dodaných celků díla.

Praktická část rozebírá problematiku proprietárních *protokolů*, podmínek jejich vývoje a aplikace. Konstatuje, že proprietární *protokol* představuje fenomén spíše obchodní a technický, než specifický z hlediska konstrukce. Dále uvádí přehled požadavků platné technické legislativy na komunikaci BIS, tedy i aplikaci *protokolů*, ze které je jednak zřetelný trend sbližování BIS a ICT a fakt, že legislativa spíše než deklaraci konkrétních *protokolů* popisuje požadované nebo zakázané stavy BIS a jejich částí.

Nakonec sumarizuje přístup významného českého výrobce k popisované problematice a uvádí přehled standardizovaných protokolů ať užitečných, či nezbytných v praxi BIS.

Tato práce poskytuje pomoc všem kolegům v oboru, kteří potřebují ucelenou informaci o této problematice, která v našem oboru nabývá na stále větším významu.

CONCLUSION

Bachelor thesis provides a theoretical analysis of the theory and practice of communication protocols applied in security systems, their design and application. It deals with the methodology of protocol design using UML. UML belongs to the recognized standards. It is a tool for modeling the status processes and systems behavior in general. Therefore, its usefulness is not limited to the protocol design, but it extends to SW and process design in general. Both of these fields are needed and their importance increases.

The thesis combines UML general theory explained in [4] and applications for UML protocol design (see [1]), applied specific modeling tool (StarUML) available under the GNU / GPL [6], in order to achieve independence from commercial tools which can also bring certain specific differences.

Furthermore, the theoretical part explains the modeling techniques and the importance of stratification protocols. This topic has grown in importance as the security systems converges to ICT and takes over open systems. In addition, it provides important insight into perceptions of closed system communications of security systems.

There are also mentioned both conventional and trendy methods for testing protocols. The reader can create an overview of the importance of this field in terms of authentication protocols confidence not only in the development of technology, but also in their application and take-supplied units work.

The practical part analyzes the proprietary protocols, the conditions for their development and application. It notes that the proprietary protocol is a phenomenon rather commercial and technical than specific aspects of design. In addition, provides an overview of the technical requirements of the legislation on communication of security systems, including application protocols. Both represent a distinct trend of the security systems and ICT convergence and the fact that the legislation rather than a specific protocol describes the required or prohibited security systems states and their parts.

At last, the thesis summarizes the approach of leading Czech manufacturer to the discussed problems and provides an overview of standardized protocols either useful or necessary in security systems practice.

This thesis was written with the desire to help all colleagues who need comprehensive information on this issue, which is becoming even more important in our industry.

SEZNAM POUŽITÉ LITERATURY

- [1] POPOVIC, Miroslav. Communication protocol engineering. Boca Raton: Taylor, 2006, ISBN 9780849398148.
- [2] *An Internet Encyclopedia: Programmed Instruction Course* [online]. [cit. 2013-05-04]. Dostupné z: <http://www.freesoft.org/CIE/Course/Section3/7.htm>
- [3] Applications of UML. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013, 6 March 2013 [cit. 2013-05-04]. Dostupné z: http://en.wikipedia.org/wiki/Applications_of_UML
- [4] KRAVAL, Ilja, RNDr. *Úvod do sémantiky UML 1.3 podle standardu OMG s komentářem*. 19.8.2001. Lipina 100, 766 01 Valašské Klobouky: Object Consulting.
- [5] FAKHROUTDINOV, Kirill. *Uml-diagrams.org* [online]. 2012 [cit. 2013-02-10]. Dostupné z: <http://www.uml-diagrams.org/>
- [6] STARUML. StarUML – aplikace a soubory nápovědy: StarUML 5.0 Developer Guide. 2007. 5.0. Dostupné z: <http://staruml.sourceforge.net/en/>
- [7] MEYER, Steve a Larry APFELBAUM. Use-Cases Are Not Requirements. [online]. March 19, 1999, s. 13 [cit. 2013-01-20]. Dostupné z: http://www.geocities.com/model_based_testing/notreqts.pdf
- [8] ABZ slovník cizích slov. RADEK KUČERA & DAUGHTER. [online]. 2006 [cit. 2013-02-16]. Dostupné z: <http://slovník-cizich-slov.abz.cz>
- [9] Proprietární software. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012, 10. 2. 2012 [cit. 2013-02-16]. Dostupné z: http://cs.wikipedia.org/wiki/Propriet%C3%A1rn%C3%AD_software.
- [10] PROTOCOL LAYERING. GREBE, David L. APOGEE LABS, Inc. [online]. [cit. 2012-12-15]. Dostupné z: <http://apogeelabs.com/pdf/Layering.PDF>
- [11] ITU-T Rec. X.200 (1994 E). Data Networks and Open system communication: Open system Interconnection – model and notation. 07.1994. Dostupné z: <http://www.itu.int/rec/T-REC-X.200-199407-I/en>
- [12] ISO/IEC 7498-1:1994. Informační technologie – Propojení otevřených systémů: Základní referenční model – Základní model. 1.3.1997. ČNI. *Poznámka: Protože*

[12] kompletně cituje [11] (kromě národních příloh), je jako zdroj uváděno [11]. Z [12] bylo převzato české názvosloví.

- [13] LINGER, Richard C. a Carmen J. TRAMMELL. Cleanroom Software Engineering Reference Model. Version 1.0. Pittsburgh, PA 15213: Carnegie Mellon University, 135 s. Dostupné z: <http://www.sei.cmu.edu/reports/96tr022.pdf>
- [14] GALLIER, Jean. Logic For Computer Science: Foundations of Automatic Theorem Proving [online]. University of Pennsylvania, 2003 [cit. 2012-12-26]. REVISED ON-LINE VERSION (2003). Dostupné z: <http://www.cis.upenn.edu/~jean/gbooks/logic.html>
- [15] CSI5118 (COMP 5302): AUTOMATED VERIFICATION AND VALIDATION OF SOFTWARE. SOMÉ, DR., Stéphane S. [online]. University of Ottawa [cit. 2012-12-26]. Dostupné z: site.uottawa.ca/~ssome/Cours/CSI5118/Integration.pdf
- [16] LYU, Michael R. Software Reliability Engineering: A Roadmap. In: [online]. Computer Science and Engineering Department The Chinese University of Hong Kong [cit. 2012-12-26]. Dostupné z: <http://ieeexplore.ieee.org>
- [17] Software Reliability Growth Models. In: WOOD, Alan. Hewlett-Packard journal: technical information from the laboratories of Hewlett-Packard Company [online]. Cupertino, CA: Tandem Computers, 1996 [cit. 2012-12-26]. Dostupné z: <http://www.hpl.hp.com/techreports/tandem/TR-96.1.pdf>
- [18] An Overview of OSI Conformance Testing: Formal Methods & Tools group. In: Casiopea [online]. University of Twente, 2001 [cit. 2012-12-26]. Dostupné z: <http://people.cs.aau.dk/~kg1/TOV03/iso9646.pdf>
- [19] KŘIVÝ, Ivan. Ostravská univerzita v Ostravě: Přírodovědecká fakulta UO [online]. Ostravská univerzita v Ostravě. Ostrava, 2005 [cit. 2012-12-27]. Dostupné z: http://prf.osu.cz/doktorske_studium/dokumenty/Random_Processes.pdf
- [20] ČSN EN 50131-1 ed.2. Poplachové systémy: Poplachové zabezpečovací a tísňové systémy; Část 1 – Systémové požadavky. ČNI, 10.2006.
- [21] ČSN EN 50131-2-2 Poplachové systémy: Poplachové zabezpečovací a tísňové systémy; Část 2.2: Detektory narušení – Pasivní IR detektory. ČNI, 12.2008.
- [22] ČSN EN 50131-5-3 Poplachové systémy: Poplachové zabezpečovací a tísňové systémy; Část 5.3: Požadavky na zařízení využívající bezdrátové propojení. ČNI, 03.2008.

- [23] ČSN EN 50133-1. Poplachové systémy – Systémy kontroly vstupů pro použití v bezpečnostních aplikacích: Část 1: Systémové požadavky. ČNI, 3.2001.
- [24] ČSN EN 50132-1. Poplachové systémy – CCTV sledovací systémy pro použití v bezpečnostních aplikacích: Část 1: Systémové požadavky. ČNI, 11.2010.
- [25] ČSN EN 50132-5. Poplachové systémy – CCTV sledovací systémy pro použití v bezpečnostních aplikacích: Část 5: Přenos videosignálu. ČNI, 4.2002.
- [26] ČSN EN 50132-5-1. Poplachové systémy – CCTV dohledové systémy pro použití v bezpečnostních aplikacích: Část 5-1: Video přenosy – obecné provozní požadavky. ČNI, 9.2012.
- [27] ČSN EN 50132-5-2. Poplachové systémy – CCTV dohledové systémy pro použití v bezpečnostních aplikacích: Část 5-2: IP video přenosové protokoly. ČNI, 9.2012.
- [28] ČSN EN 54-13. Elektrická požární signalizace: Část 13: Posouzení kompatibility komponentů systému. ČNI, 12.2005.
- [29] ČSN EN 54-25. Elektrická požární signalizace: Část 25: Komponenty využívající rádiové spoje. ČNI, 1.2009.
- [30] ČSN EN 54-2. Elektrická požární signalizace: Část 2: Ústředna. ČNI, 2.1999.
- [31] ČSN EN 54-21. Elektrická požární signalizace: Část 21: Poplachová a poruchová přenosová zařízení. ČNI, 1.2007.
- [32] ČSN EN 50134-1. Poplachové systémy – Systémy přivolání pomoci: Část 1: Systémové požadavky. ČNI, 3.2003.
- [33] ČSN EN 50134-2. Poplachové systémy – Systémy přivolání pomoci: Část 2: Aktivační zařízení. ČNI, 4.2001.
- [34] ČSN EN 50134-3. Poplachové systémy – Systémy přivolání pomoci: Část 3: Místní jednotka a kontrolér. ČNI, 3.2002.
- [35] ČSN EN 50134-5. Poplachové systémy – Systémy přivolání pomoci: Část 5: Propojení a komunikace. ČNI, 7.2005.
- [36] ČSN EN 50136-1. Poplachové systémy – Poplachové přenosové systémy a zařízení: Část 1: Obecné požadavky na poplachové přenosové systémy. ČNI, 10.2012.

TABULKA 2 až TABULKA 7: zdroj: [4], [6]

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

BIS	Bezpečnostní informační systém(y)
CID	Contact ID protokol
DB	databáze
diagram	Bez přívlastku je chápán v kontextu názvu diagramu dané části dokumentu.
DPPC	Dohledové, přijímací, poplachové centrum
DRZ	Diagram rozmístění zdrojů (Deployment diagram)
DU	Obecná datová jednotka (Data unit). Libovolný datový útvar konečné délky (paket, rámec, zpráva, datagram, apod.).
EIA	Electronics Industry Association
ExSDU	Expedited service data unit
I&HAS	Intruder and Holdup Alarm Systems. Anglický originál CZ ekvivalentu PZTS - poplachové zabezpečovací a tísňové systémy.
I/O	Vstup a výstup (obecný, logický)
ICT	Information and communications technology
ID	Obecný identifikátor, zpravidla unikátní číslo specifického formátu.
2	International Electrotechnical Commission
IETF	Internet Engineering Task Force
ISO	International Standard Organization
KA	Konečný automat (Finite State Machine - FSM)
N/ACK	Negative / acknowledgement
OEM	Original Equipment Manufacturer (díly a výrobky určené pro další výrobce)
OS	Operační systém
OSI	Open Systém Interconnection (propojení otevřených systému)
PCI	Protocol Control Information (data řízení komunikace)
PDU	Protocol data unit (DU (N)-protokolu)
PKP	Proprietární komunikační protokol
protokol	Též „komunikační protokol“
QoS	Quality of Service (kvalita služby)
SAP	Service Access Point (přístupový bod služby)
SDU	Service data unit (DU přenosové (N)-služby, obsahuje PDU)
UML	Unified Modeling Language (Unifikovaný modelovací jazyk)

SEZNAM OBRÁZKŮ

Obr. 1: Obecný postup aplikace komunikačního protokolu	8
Obr. 2: Příklad formátu zasílané zprávy - IP paket v.4.....	9
Obr. 3: fáze konstrukce komunikačního protokolu	10
Obr. 4: Struktura diagramů modelování protokolů v modelu UML.....	14
Obr. 5: diagram případu užití pro detektor zasílající měřená data sítí.....	21
Obr. 6: Diagram spolupráce pro detektor zasílající měřená data sítí.....	23
Obr. 7: Diagram tříd konečného automatu	26
Obr. 8: Diagram tříd knihovny FSM pro vývoj komunikačních protokolů.....	27
Obr. 9: Objektový diagram KA	29
Obr. 10: Sekvenční diagram komunikace detektoru.....	31
Obr. 11: Diagram aktivit detektoru.....	36
Obr. 12: Příklad využití paměťového stavu mělké historie v kompozitním stavu	39
Obr. 13: Stavový diagram detektoru.....	39
Obr. 14: DRZ topologie měřicího systému se třemi detektory a řídicím SW	42
Obr. 15: DRZ SW komponent měřicí soustavy.....	42
Obr. 16: příklad Markovova řetězce	61
Obr. 17: Schéma procesu dokumentace a zvyšování spolehlivosti SW, zdroj: [14]	62
Obr. 18: Schéma priorit výběru konstrukční varianty	68

SEZNAM TABULEK







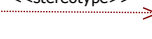
Tabulka 1: Přehled UML diagramů pro modelování komunikačních protokolů.....	13
Tabulka 2: Základní objekty diagramu případu užití.....	20
Tabulka 3: Základní prvky diagramů tříd	24
Tabulka 4: Základní prvky sekvenčních diagramů	30
Tabulka 5: Základní prvky diagramu aktivit	33
Tabulka 6: Základní prvky stavového diagramu	37
Tabulka 7: Základní prvky diagramu rozmístění komponent.....	41
Tabulka 8: Příklady axiomálních vzorců FSM a stavového stroje:	59
Tabulka 9: Příklad vývoje případových testů	61
Tabulka 10: Model zásobníku protokolů JA.....	84


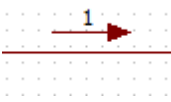


SEZNAM PŘÍLOH

PŘÍLOHA 1	VYBRANÉ VLASTNOSTI UML OBJEKTŮ	94
PŘÍLOHA 2	VYBRANÉ VLASTNOSTI UML OBJEKTŮ	96
PŘÍLOHA 3	VRSTVY OSI MODELU A JEJICH VLASTNOSTI	97
PŘÍLOHA 4	STANDARDIZOVANÉ PROTOKOLY V PRAXI BIS.....	100

PŘÍLOHA 1

VYBRANÉ VLASTNOSTI UML OBJEKTŮ

#	Relace	Popis
1.	 Asociace (Association)	<p>Vztah mezi dvěma a více <i>klasifikátory</i> při jejich vzájemném použití. Obsahuje min. 2 instance <i>AssociationEnd (Konec)</i>, (potomek <i>ModelElement</i>), který je přímou objektovou referencí a spojuje <i>asociaci s klasifikátorem</i> hodnotou atributu <i>EndParticipant</i>.</p> <p>Relace oba <i>konce</i> propojuje, každý vidí svůj <i>klasifikátor</i> (typ).</p> <p>Běžná <i>asociace</i> má všechny <i>konce</i> s <i>aggregation</i> = none</p>
2.	 0..1 1  0..* 1..* Asociace s Agregací a Kompozicí	<p>Vztah celek/část. Jen jeden Konec Asociace může být agregací - atribut <i>aggregation</i> dává tuto vlastnost celé Asociaci. Má hodnoty:</p> <ul style="list-style-type: none"> • <i>Aggregate</i>: <i>konec asociace</i> vstupuje do agregace, tj. „je součástí“. <p>Agregovaná část může být obsažena i jako část jiného celku (tato část může agregovat i jinde).</p> <ul style="list-style-type: none"> • <i>Composite</i>: „silnější“ <i>agregace</i>, druhý <i>konec asociace</i> je také část celku; daný <i>prvek</i> nemůže být součástí nikde jinde, bez něj postrádá celek smysl. <p>Na symbolech je zobrazen <i>atribut multiplicity</i>: počet koncových <i>instancí</i>, které mohou být napojeny k jedné <i>instanci</i> na daném <i>konci</i>.</p>
3.	 Generalizace (Generalization)	<p>Indikuje vztah dědičnosti <i>prvků</i>, které ji vykazují (potomci třídy <i>Generalizable element</i>). Předchůdce předává své vlastnosti a funkcionality následovníkovi, který přidá specializaci („něco navíc“).</p> <p>Vztah dědičnosti se ustanovuje mezi potomkem třídy <i>GeneralisableElemnt</i> a relací <i>Generalizace</i>, ne mezi předchůdcem a následovníkem. Ty produkuje až relace <i>Generalizace</i>.</p> <p>Šipka směřuje od předchůdce (rodič) k následovníkovi.</p>
4.	 <<include>> Začlenění (Include)	<p>Jeden prvek <i>případ užití</i> začleňuje (obsahuje beze zbytku) druhý. <i>Případ užití</i> na straně relace <i>base</i> „používá“ <i>případ užití</i> na straně <i>addition</i>. Šipka je na straně <i>base</i>.</p>
5.	 <<extend>> Rozšíření (Extend)	<p>Obdobně jako <i>include</i>, který přebírá chování <i>addition</i> strany bezpodmínečně. Extend definuje podmínku za které, extenze nastane. Ta je vyjádřena v atributu <i>Condition</i> této relace.</p> <p>Body, kde se chování rozšiřuje, jsou <i>body rozšíření</i> (<i>ExtensionPoint</i>). Každá relace Extend vidí {1..N} <i>bodů rozšíření</i>.</p>
6.	 <<stereotype>> Závislost (Depencency)	<p>Orientovaný vztah, ukazuje, že některý <i>prvek</i> nebo množina <i>prvků</i> vyžaduje nebo závisí na jiném z hlediska specifikace nebo implementace. Je to vztah dodavatele a klienta a klient je v nějakém smyslu nekompletní, sémanticky nebo strukturně závislý na dodavateli. Změna dodavatele může mít vliv na klienty.</p> <p><i>Stereotyp</i> definuje, co poskytuje dodavatel klientovi“. Definuje se několik základních stereotypů: {<i>send</i>, <i>signal</i>, <i>uses</i>, <i>call</i>, <i>bind</i>} a další.</p>

#	Relace	Popis
7.	 Spojení Instancí (Link)	<p>Potomek: <i>Asociace</i></p> <p>Propojuje dvě <i>Instance (objektů)</i>, je konkretizací (<i>instancí Asociace</i>). Obsahuje min. dva <i>Konce Spojení (LinkEnd)</i> jako <i>kompozici</i>. Ty odpovídají <i>Koncům Asociace</i>, ze kterých pocházejí. <i>Konec Spojení</i> vidí jednu <i>Instanci</i>, která jej reprezentuje.</p>
8.	 Stimul (Stimulus)	<p>Potomek: <i>Asociace</i></p> <p>Definuje komunikaci mezi dvěma <i>Instancemi</i> (např. Signál, vyvolání Operace apod.) podél jednoho <i>Link</i>, (<i>Asociace</i> na metatřídě <i>Link</i> s rolí <i>communication</i>). Jedna <i>Instance</i> je odesílatel, druhá příjemce.</p> <p>Má <i>Akci</i> s rolí <i>dispatchAction</i>, která vyvolává daný <i>Stimul</i>.</p> <p>Zvláštním případem je <i>SelfLink</i>, který definuje vnitřní <i>komunikaci Instance</i> daného <i>objektu</i></p>
9.	 Realizace (Realization)	<p>Specializovaná Relace mezi dvěma množinami <i>prvků</i> (např. <i>třídami</i>). Jeden je specifikací (dodavatel) a druhý implementací (klient). Není exaktně specifikována v UML, používá se ve složitějších případech k modelování kontextu.</p>
10.	 Přechod (Transition)	<p>Potomek: <i>Prvek modelu</i></p> <p>Směřovaný vztah mezi zdrojem a cílem <i>stavovým vrcholem (StateVertex)</i>. Může nést aktivitu definovanou v kolekci <i>Effect</i> a <i>trigger</i>, specializovaný podle <i>Události</i> - viz kap. 2.5.4.</p> <p>Zvláštní případem <i>přechodu</i> je <i>SelfTransition</i>, který reprezentuje přechody vnitřních <i>stavů objektu</i> (rekurze).</p>

PŘÍLOHA 2

VYBRANÉ VLASTNOSTI UML OBJEKTŮ

Název vlastnosti	Popis
Name	Jméno <i>prvku</i> v modelu
Visibility	Viditelnost <i>prvku</i> modelu vzhledem k danému <i>Pojmenovanému prostoru (Namespace)</i> , resp. Klasifikátoru. Nabývá hodnot: Public: viditelnost bez omezení Protected: libovolný potomek <i>prvku</i> vidí daný <i>prvek</i> Private: daný <i>prvek</i> je viditelný pouze uvnitř <i>prvku modelu</i>
Classifier	Odkaz na prvek modelu, obecně nesoucí nějakou informaci o struktuře a chování, jehož vlastnosti daný prvek s touto vlastností nabývá
Operations	Služby, které lze vyžadovat po nějakém objektu, k docílení požadovaného chování.
Attributes	Kolekce obecných vlastností <i>prvku</i> . Každý atribut nabývá dalších vlastností.
IsAbstract	PRAVDA: <i>prvek</i> je na úrovni zobecnění, že nemá smysl zavádět jeho instance. V tomto případě se instance musí objevit až v některém z potomků tohoto prvku.
IsRoot	PRAVDA: <i>prvek</i> je vrcholovým prvkem ve vztazích <i>Generalization</i> (nemá předka, tj. vyšší zobecnění).
IsSpecification	
IsLeaf	PRAVDA: daný <i>prvek</i> již nemá ve vztahu <i>Generalization</i> potomka, je posledním ve stromu. Jinak je potomka povoleno zavést.
IsActive	PRAVDA: objekt z Třídy je v samostatném threadu. NEPRAVDA: objekt je v threadu jiného aktivního objektu v rámci volajícího objektu.

PŘÍLOHA 3 VRSTVY OSI MODELU A JEJICH VLASTNOSTI

Vrstva		Funkce
(7) Aplikační	↓	Spolupracující propojené entity (zapouzdřený aplikační proces) Notace DU nezávislá na technice kódování potřebné k reprezentaci
	↓	Přímo poskytuje aplikačním procesům všechny potřebné služby a funkce přístupu ke komunikaci, které se nevyskytují v nižších vrstvách (vč. směrování, převádění, propojování) Určuje QoS, identifikaci, identifikaci abstraktních syntaxí, stanovuje stupeň zabezpečení, režim spojení (spojovaná/nespojovaná služba) Podporuje konverzi spojovaného a nespojovaného režimu
(6) Prezentační	↑	Poskytuje aplikačním entitám nezávislost na konkrétní syntaxi Transformace abstraktní syntaxe do přenosové (prezentační kontext) Omezuje velikost datové jednotky, neposkytuje segmentování a skládání
	↓	Výběr a dohodnutí vzájemně přijatelných přenosových syntaxí (spojovaný), určení přenosové syntaxe (nespojovaný) Formální specifikace a konkrétní reprezentace přenášených dat Kódování, komprese
	↓	Určuje oprávnění výkonu relačních služeb, transparentní synchronizační body relace, mapují prezentační adresy na relační (1:1, N:1).
(5) Relační	↑	V nespojovaném režimu neposkytuje segmentování a skládání, mapuje 1:1 transportní přenos Vyjednává parametry relačního spojení
	↓	Organizace a synchronizace komunikace prezentačních entit, přenos dat z klávesnice Na žádost 6. vrstvy zřizuje a uvolňuje relační spojení, nebo při výjimkách Přenos spěšných dat, řízení výjimek, oznamování chyb
	↓	Mapuje transportní adresy na relační Transportní řízení toku dat
(4) Transportní	↑	Společná obsluha relačních entit (efektivita) Překlad relačních adres na transportní (1:1) Přebírá, vyjednává a kontroluje dodržování požadavků na QoS od 5. vrstvy
	↓	SPOJOVANÁ: transparentní přenos mezi relačními entitami (spolehlivost, efektivita, ekonomika, originální pořadí DU). Zajišťuje fáze: zřizování spojení - přenos dat - uvolňování (aktivní s přenosem dat). Udržování a pozastavení spojení (více nezávislých spojení mezi párem transportních adres), Všechny <i>protokoly</i> jsou koncové (transportní asociace) NESPOJOVÁ: neposkytuje segmentování a skládání, omezení rozměrem DU

Vrstva		Funkce
		transportního protokolu Přenos spěšných dat, detekce a zotavení chyb, konverze spojovaného a nespojovaného režimu na koncových bodech
	↓	SPOJOVANÁ: výkonově optimalizuje dostupnost síťové služby (řízení toku dat, multiplexování nebo rozštěpení na síťovou vrstvu, QoS) Závislost QoS na QoS síťové vrstvy Mapování transportních adres na síťové adresy transportních entit
(3) Síťová	↑	Mapování transportních adres na síťové Vyjednává QoS, kompatibilita požadavků ve všech koncových bodech VOLITELNÉ SLUŽBY: uvolnění spojení (bez záruky doručení dat), reset, přenos spěšných dat, příjem potvrzení (od transportní entity) Integrita přepravovaných DU Abstrakce architektury: nezávislost transportních entit na směrování, propojování a převádění, na architektuře sítě a použitých technologiích. Transparentní přenos DU transportních entit po síťových spojeních v originálním pořadí (řazení) NESPOJOVANÁ: přenos DU definované velikosti, QoS, oznamování chyb
		Konverze spojovaného a nespojovaného režimu Přenos dat mezi libovolnými dvěma síťovými entitami (tandemové a paralelní sítě a podsítě) Zřízení, udržování síťových spojení směrování, převádění a (sériové) propojování sítě a podsítí (mezilehlé otevřené systémy), multiplexování (vč. propojených tandemových sítí), segmentování, blokování, řízení toku, detekce chyb a zotavení, přenos spěšných dat, reset, výběr služeb Dělení síťové vrstvy do podvrstev v závislosti na propojování podsítí
	↓	Služby skokových propojení Řízení propojení datových okruhů, využití spojení 2. vrstvy
(2) Spojová (Linková)	↑	Propojení datových okruhů 1. vrstvy, identifikátory datových okruhů NESPOJOVANÁ: spojové adresy, přenos DU konečné velikosti, QoS SPOJOVANÁ: spojové spojení, identifikátory koncových bodů spojových spojení, oznamování chyb, reset, řízení toku
		Přenos DU, rozštěpení spojení, identifikace DU 1. vrstvy (bitové separátory), řazení, detekce a zotavení chyb
	↓	Konverze služby na ne/spojovaný režim Směrování, převádění, propojování Detekce a oprava chyb 1. vrstvy Využívá fyzická spojení
(1) Fyzická	↑	Služby ke spolupráci aplikačních entit Řízení propojení a převádění datových okruhů (aktivace a deaktivace)

Vrstva		Funkce
		<p>spojení), identifikátory koncových bodů a datových okruhů.</p> <p>Nerozlišuje spojovaný a nespojovaný režim</p> <p>Datové jednotky (rámce), řazení, detekce a oznamování chyb, QoS</p>
	I	<p>Transparentní přenos bitových toků mezi spojovými entitami přes fyzické médium v simplexním a polo/duplexním resp. synchronním a asynchronním režimu</p> <p>Multiplexování (využití média pro různé okruhy a <i>protokoly</i>)</p> <p>Dvou a vícebodové datové okruhy (fyzická propojení koncových entit), převáděcí systémy (mezilehlé otevřené systémy) se sériovým nebo paralelním přenosem</p>
Fyzické médium	↑	Fyzické a fyzikální vlastnosti (mechanické, elektromechanické, elektromagnetické atd.)

PŘÍLOHA 4 STANDARDIZOVANÉ PROTOKOLY V PRAXI BIS

Protokol	Standard	Popis funkce	Vrstva
RS-232-C	EIA 232 C ITU-T V.24	Asynchronní sériová komunikace. Log. 0 a 1 reprezentují bipolární úrovně napětí a dle zařízení mají ± 5 V, ± 10 V, ± 12 V nebo ± 15 V. Sběrnice má 3 základní vodiče (příjem RxD, vysílání TxD a společná zem GND). Doplnují je vodiče k řízení přenosu (vstupy DCD, DSR, CTS, RI, výstupy DTR, RTS). Ty mohou a nemusí být používány. Pořadí přenosu datových bitů je od nejméně významného bitu (LSB) po bit nejvýznamnější (MSB). Délka rámce je volitelná, obvykle se používá 8 bitů. V praxi BIS se používá jako systémová sběrnice pro komunikaci konfigurovaných zařízení (např. ústředna I&HAS a PC) a ke komunikaci systémových komponent, nejčastěji přijímačů DPPC s automatizačním SW.	L1
RS-485	EIA 485 ITU-T V.11	Standard sériové komunikace pro průmyslové prostředí. Vytváří vícebodové sběrnice pro max. 32 zařízení s možností komunikace do 1200 m. Přenosová rychlost do 10 m vzdálenosti je až 10 Mb/s. Při komunikaci na delší vzdálenost se vedení zakončuje odpory, které zabrání odrazům signálu na koncích vedení. Rozlišuje se dvojvodičová (polo duplexní) a čtyřvodičová (plně duplexní) verze. Základním vodičem je kroucený pár. Používá 7 nebo 8 bitové rámce, start bit, 1 a více stop bitů, případně paritní bit. Start bit reprezentuje log. 0, stop bit a neaktivní stav log. 1. V praxi BIS se uplatňuje zejména jako vnitřní, proprietární, systémová sběrnice komponentů I&HAS, EPS, SKV atd. Nezanedbatelné výhody představují vysoká odolnost proti rušení a možnost větvení.	L1
G.703	ITU-T G.703	Standard pulsní kódové modulace (PCM) pro přenos hlasu a dat prostřednictvím digitálních linek (např. T1 a E1 - metalické vedení, obvyklé ve veřejné telefonní síti). V Praxi BIS představuje obecnou datovou konektivitu v současnosti využívanou většinou v ISDN přípojkách o 30 kanálech. Ty nacházejí kromě běžné telefonní komunikace uplatnění zejména ve sběru dat na DPPC z BIS v objektech komunikujících pevnou telefonní linkou.	L1
M-Bus	EN 13757-2, 3	Průmyslový protokol pro aplikace s požadavkem vysoké odolnosti proti rušení. Komunikace probíhá v režimu Master-Slave. Master komunikuje změnou napětí 36 V (log. 1) a 24 V (log. 0). Terminály (Slave) komunikují změnou proudu; v klidovém stavu odebírá 1,5 mA (log. 1), pro log. 0 je proud o 11-20 mA vyšší. Asynchronní sériová komunikace s rámcem 8b. Adresy stanic mají 1B (0-250), lze použít sekundární adresy (8B) implementované v síťové vrstvě, pak je primární adresa 253.	L2

Protokol	Standard	Popis funkce	Vrstva
		<p>Na sběrnici může být připojeno 250 stanic. Délka segmentu je 1000 m při rychlosti 300 bps nebo 350 m při rychlosti 9.600 bps. Sběrnice zároveň zajišťuje napájení terminálů.</p> <p>U některých BIS systémů se M-Bus aplikuje jako alternativa vnitřní systémové komunikace k protokolu RS-485.</p>	
Wi-Fi	IEEE 802.11	<p>Standard bezdrátové komunikace (WLAN) pro lokální bezdrátové sítě v pásmu generálního povolení. Určen k propojování počítačových sítí a vytváření ad-hoc sítí. Hojně využíván k budování „poslední míle“ mezi poskytovatelem připojení Internetu a uživatelem.</p> <p>V BIS se uplatňuje nepřímo, ale široce. Kromě záložních připojení libovolných technologií k Internetu je nutné brát v úvahu fakt, že velká část domácností a malých firem (SOHO) tuto technologii využívá jako hlavní a jedinou přípojku k Internetu. BIS technologie využívající IP komunikaci (připojení k DPPC, přenos videa) pak pracují na této technologii.</p>	L2
ZigBee	IEEE 802.15.4	<p>Spojení nízko výkonových zařízení v sítích PAN na vzdálenosti do 75 m. Díky ad-hoc směrování přes libovolný node umožňuje komunikaci i na větší vzdálenosti, bez přímé radiové viditelnosti jednotlivých zařízení. Aplikace v průmyslu a sensorových sítích. Pracuje v pásmech generálního povolení 868 MHz, 902-928 MHz a 2,4 GHz. Přenosová rychlost 20, 40, 250 kbit/s.</p> <p>V některých BIS aplikacích slouží jako nosná platforma bezdrátové komunikace komponent, např. i v hybridních systémech, integrujících BIS a (malou) automatizaci.</p>	L2
Ethernet	IEEE 802.3	<p>Kromě jiného představuje standardizované L2 rozhraní ICT technologií, které se stále více prosazuje i v technologiích BIS, protože v současnosti představuje standard datové přípojky a rozvodů v mnoha budovách. Pracuje s topologií hvězdy a typicky využívá kroucenou dvojlinku o čtyřech párech. Na úrovni přenosu dat pracuje s rámci, jejich délka se liší podle verze protokolu (např. 10Base-2, 100BASE-T4, 100BASE-T2, 1000BASE-T atd.).</p> <p>V BIS se stává standardem pro komunikaci systémů s DPPC a v CCTV systémech pro IP kamery. U rozsáhlejších systémů SKV a EPS se lze setkat s propojením řídicích modulů v rámci distribuované architektury a dále např. u biometrických čteček.</p>	L2
DSS1	ITU-T I.411	Také protokol D kanálu ISDN. Poskytuje služby jako je ANI (CLIP) a DNIS, což jsou identifikace volajícího a volaného čísla, využívané v praxi příjmu a zpracování zpráv v DPPC.	L2
Q.23	ITU-T	Definice DTMF (dual tone multifrequency signaling), tj. tónové volby. Definuje frekvenční páry pro 12 pozic	L2

Protokol	Standard	Popis funkce	Vrstva
	Q.23	<p>alfanumerické klávesnice telefonu a jejich význam. Související doporučení je Q.24.</p> <p>V BIS se využití tónové volby uplatňuje poměrně široce. Zahrnuje dálkové ovládání a nastavování BIS systémů, prostřednictvím telefonní linky, přístupovou identifikaci zadáním alfanumerického hesla at' při přístupu k BIS nebo do DPPC a v neposlední řadě je v proprietární podobě (změněný signalizační standard co do délky a frekvencí tónů) využíván k signalizaci zpráv přenášných z BIS na DPPC. Vzhledem k tomu, že v současné době je stále většina - zejména - I&HAS vybavena komunikátory pro analogovou telefonní linku, představuje tento standard významný prvek i při digitalizaci přenášných zpráv a jejich komunikaci na L5, využitím odlišných transportních systémů.</p>	
PPPoE	RFC 2516	<p>Point-to-Point Protocol over Ethernet. Protokol pro zapouzdření PPP rámců do rámce Ethernet. Používá se u internetových služeb na bázi xDSL. Představuje řešení k tunelování paketů DSL připojením k IP síti a odtud do zbytku Internetu. Je navržen pro malé LAN s jednotlivými nezávislými připojeními k internetu.</p> <p>Z toho důvodu představuje významný linkový protokol při komunikaci domácností a malých firem (SOHO) s DPPC a dalšími distribuovanými částmi BIS.</p>	L2
ARP RARP	RFC 826 RFC 903	<p>Address Resolution Protocol. Používá se v IP sítích k získání MAC adresy jiného zařízení z jeho IP adresy, když se mají na L2 odeslat data adresátovi, o němž odesílatel zná jen IP adresu. Odesílatel pošle dotaz s hledanou IP adresou a současně dává na vědomost vlastní IP a MAC adresu. Dotaz posílá broadcastem (MAC adresa sítě - pro Ethernet FF:FF:FF:FF:FF:FF). Všechna zařízení dotaz obdrží a uloží údaje o odesílateli do své ARP tabulky. Zařízení s hledanou IP adresy odpoví. Informace v ARP tabulkách jsou uloženy do vypršení platnosti, používají se opakovaně. RARP protokol pracuje reverzně - ze známé MAC adresy získává IP adresu.</p> <p>V prostředí se zvýšenými nároky na bezpečnost je výhodnější použít statickou tabulku přiřazení MAC a IP adres; místo skutečného nositele hledané IP adresy může odpovědět útočník a stáhnout tak data určená nositeli. Z hlediska BIS jde o obecný síťový protokol, jehož vlastnosti je třeba znát z důvodů uvedených výše. Při chybně nastavených parametrech ARP tabulek může zatěžovat síť nepřiměřeným provozem.</p>	L2
ISDN	ITU-T I.431	<p>Množina signalizačních standardů, pracujících od L1 a po L3, pro souběžný přenos dat, videa, hlasu a dalších síťových služeb, využitím metalických, spojovaných okruhů veřejných telefonních sítí. Poskytuje i přístupové rozhraní do paketových sítí. Podle konkrétní implementace obsahuje množství paralelních datových (B) kanálů, kapacity 64 kbps, kódovaných kodekem G.711, pro přenos uživatelských dat. Dále řídicí D kanál.</p>	L3

Protokol	Standard	Popis funkce	Vrstva
		Jako služba je rozšířený především pro telefonní spojení vysoké kvality a dostupnosti. Díky plnému řízení poskytuje vysokou dostupnost. V BIS nachází uplatnění především v aplikacích s potřebou velkokapacitní paralelní komunikace telefonních hovorů - tedy jako centrální přístupový bod pro komunikaci BIS s DPPC využitím telefonních linek.	
NAT	RFC 1631	<p>Způsob přístupu LAN izolované za routerem k vnějším sítím, přepisem výchozí a/nebo cílové IP adresy, případně TCP/UDP portu a změna kontrolního součtu IP, TCP/UDP IP paketů. Adresy z LAN přeloží na jedinečnou adresu, pro vstup do jiné sítě (např. Internet) a záznam o překladu uloží pro náhodně zvolený vnitřní port, na kterém komunikuje zpět do LAN. Většinou se používá pro přístup více počítačů z LAN na Internet využitím jediné veřejné IP adresy, čímž řeší zejména světový nedostatek IPv4 adres.</p> <p>Z hlediska aplikace v BIS je důležité, že některé specifické aplikace nemohou využívat NAT z důvodu požadavků přímého spojení nebo z důvodu rozpadu časování, způsobeného zpožděním při překladu. Zvyšuje však také bezpečnost počítačů izolovaných za routerem.</p>	L3
IPSec	RFC 2401, 2411	<p>Bezpečnostní rozšíření IP protokolu autentizací a šifrováním IP paketů - zabezpečení na síťové vrstvě. Poskytuje bezpečnost kterékoliv síťové aplikaci. Vytváří jednosměrné logické kanály (Security Associations), pro duplex se používají dvě SA.</p> <p>Ověřuje přijaté pakety, zda vyslaný paket odpovídají odesilateli (Authentication Header). Účastníci zvolí formu šifrování paketu, šifrují se celé pakety kromě IP hlavičky, případně celý paket a přidá se nová IP hlavička (Encapsulating Security Payload).</p> <p>Je definován několika desítkami RFC.</p>	L3
IPv4	RFC 791	<p>Datově orientovaný protokol pro síť s přepojováním paketů. Představuje nespojovanou (connectionless) přenosovou službu - negarantuje integritu přenášených dat (doručení, pořadí, vyloučení duplicit apod.). Tyto služby poskytují až protokoly transportní vrstvy. Nese pouze kontrolní součet hlavičky datagramu se služebními údaji. Umožňuje fragmentaci (dělení IP datagramů na menší části), kvůli kompatibilitě s DU nižších vrstev. Obvyklá velikost datagramu je 1,5 kB (max. délka rámce Ethernet). Některé vysokorychlostní sítě používají kvůli nižší režii větší délky a naopak tunelované spoje velikost datagramu kvůli vlastní režii snižují.</p> <p>Nese informaci TIME_TO_LIVE (TTL) - udává, jak dlouho se IP datagram může nacházet v soustavě vzájemně propojených sítí. Představuje ochranu proti nekonzistentním směrovacím tabulkám, kdy by mohlo dojít k situaci, že by</p>	L3

Protokol	Standard	Popis funkce	Vrstva
		<p>IP datagram byl směrován v kruhu, nekonečně dlouho. Každý router v síti (mezilehlá brána) snižuje TTL podle toho, jak dlouho se v ní datagram "zdrží". Datagram s TTL 0 je zahozen.</p> <p>Z hlediska BIS představuje základní a důležitý, síťové vrstvy, který zajišťuje abstrakci sítí a umožňuje tak technologicky nezávislou aplikaci konkrétních funkcí ve vyšších vrstvách.</p>	
ICMP	RFC 792	<p>Internet Control Message Protocol. Používá se k odesílání chybových zpráv (např. oznámení, o nedostupnosti požadované služby nebo zařízení, chyby doručení datagramů apod.) a dále pro účely diagnostiky a směrování (ping, tracer). Pracuje na 3. vrstvě OSI modelu. IP protokol ICMP zprávu zapouzdří novou IP hlavičkou (aby se dostala zpět k odesílateli) a datagram odešle. Nesená chybová zpráva je po přijetí doručena procesu, který vyvolal odeslání původního IP paketu.</p> <p>Z hlediska BIS nemá konkrétní specifické uplatnění, představuje však důležitý protokol síťové infrastruktury.</p>	L3
SCTP	RFC 2960 RFC 3286	<p>Stream Control Transmission Protocol. Protokol transportní vrstvy, Má několik navzájem nezávislých kanálů, které jsou přepravovány paralelně. Po navázání spojení lze přenášet řadu navzájem nezávislých streamů s garantovaným doručením. A ve správném pořadí. Výpadky v jednom proudu neovlivňují ostatní. Monitoruje všechny cesty a udržuje záznam o jejich stavu. Komunikující uzel může mít několik IP adres IPv4 a IPv6. Během komunikace je jedna primární a jí jsou posílána data. Při poruše se vybírá jiná, má-li primární adresa opakované problémy s dostupností, přejde odesílatel na jinou. Data doručuje v balících pomocí proudů; eliminuje tak chyby bloků dat, jako je tomu u TCP. Návrh jeho vnitřní struktury komplikuje některé typy DoS útoků.</p> <p>Z hlediska BIS nemá konkrétní specializaci. Je vhodný pro přenos streamově orientovaných dat a představuje alternativu k TCP.</p>	L4
RTP	RFC 3550	<p>Real-time Transport Protocol. Standard paketového doručování zvuku a videa Internetem. Často používá v systémech streamového přenosu s RTSP. Přenáší datové toky vyjednané signálními protokoly, (např. H.323 nebo SIP) a je technickým základem VoIP. Nejčastěji je zapouzdřen UDP protokolem. Poskytuje určení užitečného zatížení, číslování sekvencí, časová razítka a sledování přenosu. Jeho modifikací je zabezpečená verze - SRTP (Secure RTP) definovaný RFC 3711.</p> <p>Z hlediska BIS představuje jeden z protokolů používaným standardem přenosu videa ONVIF.</p>	L4

Protokol	Standard	Popis funkce	Vrstva
UDP	RFC 768	<p>User Datagram Protocol. Protokol transportní vrstvy orientovaný na zprávy. Neobsahuje služby garantovaného doručení, vytváří nespolehlivé, nespojované připojení (obdobně jako IP). Přidává pouze kontrolní součty a identifikaci portů. Je za to velmi jednoduchý (obsluha má 16 B) a vhodný pro aplikace na principu otázka-odpověď (např. DNS, SNMP, DHCP a RIP, sdílení souborů), dále pro servery, které obsluhují mnoho klientů a nemohou efektivně držet takový počet otevřených socketů. Dále pro multimediální aplikace, kde nemá smysl opakovaně posílat již neaktuálních nedoručených zpráv (např. VoIP - SIP, online hry apod.).</p> <p>Z hlediska BIS představuje oblíbený transportní protokol. Jednak je využíván ve standardu přenosu videa ONVIF, konkrétně pro přenos RTP/RTCP protokolu, jednak je v módu zpětného potvrzování zpráv používán pro komunikaci BIS s DPPC.</p>	L4
TCP	RFC 793	<p>Transmission Control Protocol. Plní úlohu transportní vrstvy. Umožňuje vytvořit spojově orientované spojení. Garantuje spolehlivé doručení a ve správném pořadí, umožňuje vytvořit nezávislá spojení aplikačních entit jednoho zařízení. Používá služby nespolehlivého IP protokolu opakovaným odesíláním paketů při ztrátě a správným řazením přijatých, očíslovaných paketů, zajišťuje správné doručení kontrolním součtem paketů. Provádí fragmentaci proudu vstupních dat do paketů.</p> <p>Z hlediska BIS představuje základní a důležitý, transportní vrstvy, který zajišťuje abstrakci sítí a umožňuje tak technologicky nezávislou aplikaci konkrétních funkcí ve vyšších vrstvách.</p>	L4
SIA	ANSI/SIA DC-09-2007	<p>Protokol pro komunikaci BIS a DPPC. Původní deklarace (SIA DC-03-1990.01 (R2003.10)) definuje protokol pro komunikaci využitím telefonní linky. Hlavní rozdíl od popsaného protokolu CID spočívá v tom, že protokol SIA je schopen přenosu souvisejících textových popisů jednotlivých zón, uložených v paměti daného BIS a dále přenáší časové razítko signalizované události z monitorovaného BIS (stav jeho vnitřních hodin). Poskytuje tedy vyšší míru automatizace (klade menší nároky na údržbu a konfiguraci záznamu daného BIS v DPPC) a ve své teoretické deklaraci poskytuje vyšší míru bezpečnosti komunikace.</p> <p>Současná deklarace vytváří standard, který přenáší logický rámec na L5 do prostředí IP sítí, přičemž explicitně předpokládá přenos využitím některého z transportních protokolů (TCP, UDP) a zabezpečení šifrovacím standardem AES.</p>	L5
RTCP	RFC 3550	RTP Control Protocol. Řídící protokol pro distribuci zvuku a videa v reálném čase. Používá se k přenosu kontrolních paketů účastníkům streamované relace a poskytuje řídící data RTP protokolu tak, že kolektuje data o mediální session	L5

Protokol	Standard	Popis funkce	Vrstva
		(počet odeslaných a ztracených bajtů a paketů, jitter (kolísání zpoždění), zpětnou vazbu a dobu odezvy. Aplikace tato data používá k řízení QoS (např. řízení rychlosti datového toku, změny kodeku apod.). Alternativou je SRTCP (RFC 3711), který poskytuje šifrování toku a autentizaci. Z hlediska BIS představuje jeden z protokolů využívaných standardem ONVIF.	
CID	SIA DC-05-1999.09	Ademco SIA Contact ID. Protokol navržený pro komunikaci I&HAS a DPPC využitím telefonní linky. Obsahuje signalizační rámec na L2 a logický rámec na L5. V originálním provedení využívá vlastní dvojtónové signalizace inspirované Q.23 a Q.24. V současnosti se uplatňuje jako standard, jeho logický rámec je přenášeny na další komunikační prostředí, typicky UDP/IP, TCP/IP, SMS apod.	L5
H.264		Standard pro kompresi videa. Též H.264/MPEG-4 Part 10 nebo AVC. Byl vytvořen pro přenos videa vyšší kvality při nižší přenosové rychlosti. Využívá blokově orientovanou motion kompresi - záznam rozdílů na základě referenčního obrazu. Kromě širokého mediálního nasazení si vydobyl významného postavení v aplikacích CCTV systémů, kde se používá jednak jako kodek pro přenos obrazu z IP kamer a pro přenos mezi vzdálenými pracovišti monitoringu - jak klientských aplikací CCTV systémů, tak pro přenosy na DPPC.	L6
G.711	ITU-T G.711	Kodek pro přenos hlasových frekvencí (300 - 3.400 Hz) pulsní kódovou modulací. Obvykle je zapouzdřen G.703 protokolem v pevných sítích, ale i v dalších standardech jako H.320, H.323 (přenos v paketových sítích). Zajišťuje kontinuální tok 64 kbps při vzorkovací frekvenci vstupního signálu 8 kHz. Jeho klíčovou výhodou je bezztrátová logaritmická komprese, která je nezbytným předpokladem pro korektní komunikaci při sběru dat na DPPC z BIS v objektech komunikujících pevnou telefonní linkou. Uplatňuje se i v dalších BIS, např. ve standardu ONVIF pro přenos videa.	L6
TLS SSL	RFC 5246 RFC 6101	Transport Layer Security (Secure Sockets Layer). Poskytuje komunikujícím stranám autentizaci a soukromí při komunikaci. Využívá asymetrické šifrování. Obě strany před zahájením komunikace vygenerují privátní a veřejný klíč, veřejné klíče si při zahájení komunikace vymění a ověří je pomocí jejich HASH. Ověření proběhne buď u protistrany jiným komunikačním kanálem, nebo se klíč považuje za důvěryhodný, pokud je digitálně podepsaný uznávanou certifikační autoritou, jejíž veřejný klíč je v důvěryhodném úložišti každé komunikující strany (THAWTE, VeriSign, RapidSSL, GeoTrust, apod.). Typicky je autentizován pouze server. Při vzájemné autentizaci jsou ověřovány obě strany;	L6

Protokol	Standard	Popis funkce	Vrstva
		<p>tato technika vyžaduje infrastrukturu veřejných klíčů (PKI) pro klienty.</p> <p>Každé DU je přiřazen protokol vyšší vrstvy, může být připojen autentizační MAC kód (message authentication code) a může být zašifrována. Běží mezi aplikačními protokoly a nad spolehlivým transportním protokolem (TCP, UDP apod.). Lze jej použít k tunelování všech síťových protokolů a vytvoření VPN.</p> <p>Z hlediska BIS představuje možný univerzální standard pro bezpečný přenos dat. Přímo je využíván standardem ONVIF.</p>	
SOAP		<p>Simple Object Access Protocol. Protokol pro výměnu zpráv ve formátu XML. Představuje základní vrstvu komunikace mezi webovými službami a pro složitější komunikaci. Je tvořen obálkou, která definuje obsah zprávy a jak jej zpracovat, kódovací pravidla k vyjádření instancí definovaných datových typů, a konvencí volání procedur a reakcí. Má tři hlavní charakteristiky: rozšiřitelnost, neutralita (může být nesen v kterémkoli transportním protokolu jako HTTP, SMTP, TCP apod., včetně jejich šifrovaných verzí) a nezávislosti (umožňuje jakéhokoli SW model). Transport skrze HTTP protokol představuje výhodu při procházení firewally.</p> <p>Z hlediska BIS představuje jeden z protokolů využívaných standardem ONVIF.</p>	L7
DHCP	RFC 2131	<p>Dynamic Host Configuration Protocol. Poskytuje automatizaci konfigurace IP zařízení v síti. Centralizuje a významně zjednodušuje správu zařízení v síti (např. přidávání nových zařízení, hromadné změny parametrů) a přispívá k bezpečnosti skrytím technických detailů uživateli.</p> <p>DHCP server přiděluje DHCP protokolem zařízením parametry nutné ke komunikaci IP protokolem (IP adresu, masku sítě, implicitní bránu a adresu DNS serveru). Platnost přidělených údajů je časově omezená, proto je v zařízení vyžadován DHCP klient, který platnost prodlužuje. Je zapouzdřen UDP protokolem.</p> <p>Z hlediska BIS nemá specifický význam, představuje důležitý protokol hostitelské infrastruktury.</p>	L7
RTSP	RFC 2326	<p>Real Time Streaming Protocol. Vysílá obsah jednosměrným streamem (unicast). Byl vyvinut k řízení doručování dat v reálném čase, především zvuku a videa. Je zapouzdřen v transportním protokolu s opravou chyb (TCP). Podporuje ovládací funkce přehrávače (stop, pauza, pohyb v záznamu). Jde o protokol řízení, používá se společně s protokolem RTP. Vyjedná nejlepší způsob doručování obsahu a nasměruje protokol RTP na doručení obsahu streamu.</p> <p>Z hlediska BIS představuje jeden z protokolů využívaných standardem pro přenos videa ONVIF.</p>	L7
HTTPS	RFC 2818	<p>Hypertext Transfer Protocol Secure. Nadstavba protokolu HTTP, umožňuje zabezpečené spojení mezi klientem a</p>	L7

Protokol	Standard	Popis funkce	Vrstva
		<p>serverem (na rozdíl od HTTP protokolu, který je otevřený) a umožňuje ověřit identitu protistrany. Používá HTTP a přenášená data šifruje (ochraňuje) protokolem SSL nebo TLS. Standardní port serveru je 443. Druhou technikou aplikovanou pro HTTPS po šifrování je autentizace obou komunikujících stran, obvykle technikou ověření veřejného klíče certifikátu, vydaného důvěryhodnou certifikační autoritou.</p> <p>Z hlediska BIS představuje protokol pro zabezpečenou komunikaci mezi klientem a serverem. Např. pro přenos hostovaných dat apod.</p>	
SSH	RFC 4251	<p>Secure Shell. Zabezpečený protokol. Náhrada za TELNET, RLOGIN, RSH a další protokoly, které posílají nechráněná hesla a umožňují jejich zachycení při přenosu sítí. Poskytuje bezpečnou komunikaci, která se využívá pro přístup k příkazovému řádku, resp. pro obecný přenos dat využitím síťového tunelování. Poskytuje autentizaci účastníků komunikace, transparentní šifrování dat, garantuje jejich integritu a poskytuje volitelnou bezztrátovou kompresi. Je dostupný ve verzích SSH-1 a SSH-2. Mechanismus autentizace je principiálně shodný jako u TLS.</p> <p>Z hlediska BIS představuje možný univerzální standard pro bezpečný přenos dat. Kromě užitečných funkcí však (stejně jako ostatní šifrované protokoly) může představovat bezpečnostní riziko v podobě vytváření parazitních vstupních bodů k úniku informací a k průniku do vnitřní sítě.</p>	L7
NTP	RFC 5905	<p>Network Time Protocol. Poskytuje synchronizaci časových základů zařízení v síti. Je odolný k proměnlivému zpoždění v doručování paketů. Je zapouzdřen UDP protokolem. Stanovuje čas z nepatrných odchylek odpovědí časových serverů. Používá čas UTC s příznaky pro přestupné sekundy. NTP v.4 dosahuje chyby <10 ms. V LAN může dosáhnout přesnosti až 0,2 ms. Nejlepšího výkonu dosahují zařízení, kde jádro OS řídí čas fázovým závěsem (místo přímého dosazování času do systémových hodin).</p> <p>Jednodušší forma NTP je Simple Network Time Protocol (SNTP), který neuvažuje zpoždění paketů v síti a neuchovává stav předchozí komunikace.</p> <p>Z hlediska BIS představuje důležitý protokol infrastruktury, bez něhož by nebyla možná činnost zejména distribuovaných systémů a systémů využívajících klientské stanice pro interakci s uživatelem.</p>	L7