EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Vysoká škola báňská – Technická univerzita Ostrava

Fakulta strojní

PROJEKTOVÁNÍ INFORMAČNÍCH SYSTÉMŮ

Učební text předmětu Informační systémy

Jolana Škutová

Ostrava 2010/2011

Tyto studijní materiály vznikly za finanční podpory Evropského sociálního fondu (ESF) a rozpočtu České republiky v rámci řešení projektu OP VK CZ.1.07/2.3.00/09.0147 „Vzdělávání lidských zdrojů pro rozvoj týmů ve vývoji a výzkumu“.

Název: Projektování informačních systémů
Autor: Jolana Škutová
Vydání: první, 2011
Počet stran: 86
Náklad: 10
Studijní materiály pro studijní obor Automatické řízení a inženýrská informatika Fakulty strojní
Jazyková korektura: nebyla provedena.

Tyto studijní materiály vznikly za finanční podpory Evropského sociálního fondu a rozpočtu České republiky v rámci řešení projektu Operačního programu Vzdělávání pro konkurenceschopnost.

Název: Vzdělávání lidských zdrojů pro rozvoj týmů ve vývoji a výzkumu
Číslo: CZ.1.07/2.3.00/09.0147
Realizace: Vysoká škola báňská – Technická univerzita Ostrava

ISBN <978-80-248-2766-7>

POKYNY KE STUDIU

Analýza informačních systémů

[Poznámka: vyberte jednu z variant, případně vytvořte další variantu.]

Pro předmět 2. semestru oboru Automatizační technika a inženýrská informatika jste obdrželi studijní balík obsahující:

- integrované skriptum pro distanční studium obsahující i pokyny ke studiu,
- přístup do e-learningového portálu obsahující doplňkové animacemi vybraných částí kapitol.

Prerekvizity

Pro studium této opory se předpokládá znalost na úrovni absolventa předmětu Databáze a Internet a předmětu Programování aplikací pro Internet II.

Cílem učební opory

Cílem je seznámení se základními pojmy analýzy systémů, základy jazyka UML a jeho softwarovou podporou. Po prostudování modulu by měl student být schopen provést analýzu a návrh informačního systému s podporou jazyku UML a zpracovávat ji v softwarovém prostředí.

Pro koho je předmět určen

Modul je zařazen do magisterského studia oboru Automatizační technika a inženýrská informatika studijního programu Strojní inženýrství, ale může jej studovat i zájemce z kteréhokoliv jiného oboru, pokud splňuje požadované prerekvizity.

Skriptum se dělí na části, kapitoly, které odpovídají logickému dělení studované látky, ale nejsou stejně obsáhlé. Předpokládaná doba ke studiu kapitoly se může výrazně lišit, proto jsou velké kapitoly děleny dále na číslované podkapitoly a těm odpovídá níže popsaná struktura.

Při studiu každé kapitoly doporučujeme následující postup:

Čas ke studiu: xx hodin

Na úvod kapitoly je uveden čas potřebný k prostudování látky. Čas je orientační a může vám sloužit jako hrubé vodítko pro rozvržení studia celého předmětu či kapitoly. Někomu se čas může zdát příliš dlouhý, někomu naopak. Jsou studenti, kteří se s touto problematikou ještě nikdy nesetkali a naopak takoví, kteří již v tomto oboru mají bohaté zkušenosti.

Cíl: Po prostudování tohoto odstavce budete umět

- Popsat …
- Definovat …
- Vyřešit …

Ihned potom jsou uvedeny cíle, kterých máte dosáhnout po prostudování této kapitoly – konkrétní dovednosti, znalosti.

Výklad

Následuje vlastní výklad studované látky, zavedení nových pojmů, jejich vysvětlení, vše doprovázeno obrázky, tabulkami, řešenými příklady, odkazy na animace.

Shrnutí pojmů

Na závěr kapitoly jsou zopakovány hlavní pojmy, které si v ní máte osvojit. Pokud některému z nich ještě nerozumíte, vraťte se k nim ještě jednou.

Otázky

Pro ověření, že jste dobře a úplně látku kapitoly zvládli, máte k dispozici několik teoretických otázek.

Úlohy k řešení

Protože většina teoretických pojmů tohoto předmětu má bezprostřední význam a využití v praxi, jsou Vám nakonec předkládány i praktické úlohy k řešení. V nich je hlavním významem předmětu schopnost aplikovat čerstvě nabyté znalosti pro řešení reálných situací.

Klíč k řešení

Výsledky zadaných příkladů i teoretických otázek jsou uvedeny v závěru učebnice v Klíči k řešení. Používejte je až po vlastním vyřešení úloh, jen tak si samokontrolou ověříte, že jste obsah kapitoly skutečně úplně zvládli.

Úspěšné a příjemné studium s tímto učebním textem Vám přeje Jolana Škutová.

OBSAH

1 PROJEKTOVÁNÍ INFORMAČNÍCH SYSTÉMŮ

Modelování by mělo být nedílnou součástí oblasti inženýrství, výroby a konstruování. Složité softwarové návrhy by bylo obtížné popsat slovně, proto je možné vhodné využít grafického nástroje v podobě diagramů. Modelování má tři hlavní výhody: vizualizaci, řízení komplexnosti návrhu a jednoznačnou komunikaci. UML se stává standardem pro návrh objektově orientovaných softwarů.

1.1 Modelovací jazyk UML

Čas ke studiu: 40 minut

Cíl: Po prostudování tohoto odstavce budete umět

- Definovat pojem UML.
- Popsat dvě základní skupiny diagramů.
- Rozlišovat úrovně využití UML.
- Rozvrhnout analýzu systému do čtyř základních fází v rámci unifikovaného procesu.

Výklad

UML ™ (Unified Modeling Language ™) je grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci navrhovaných systémů. UML podporuje objektově orientovaný přístup k analýze, návrhu a popisu programových systémů. UML neobsahuje způsob, jak se má používat, ani neobsahuje metodiky, jak analyzovat, specifikovat či navrhovat programové systémy. UML je pouze nástroj, který lze použít mnoha různými způsoby na podporu metodiky vývoje softwaru, ale sám o sobě nespecifikuje ani metodologii a ani postup. Je nutné podotknout, že UML je nezávislé na konkrétním implementačním prostředí a lze tento modelovací jazyk aplikovat na různé typy procesů a využít jej pro účely postupných etap vývoje modelovaného systému.

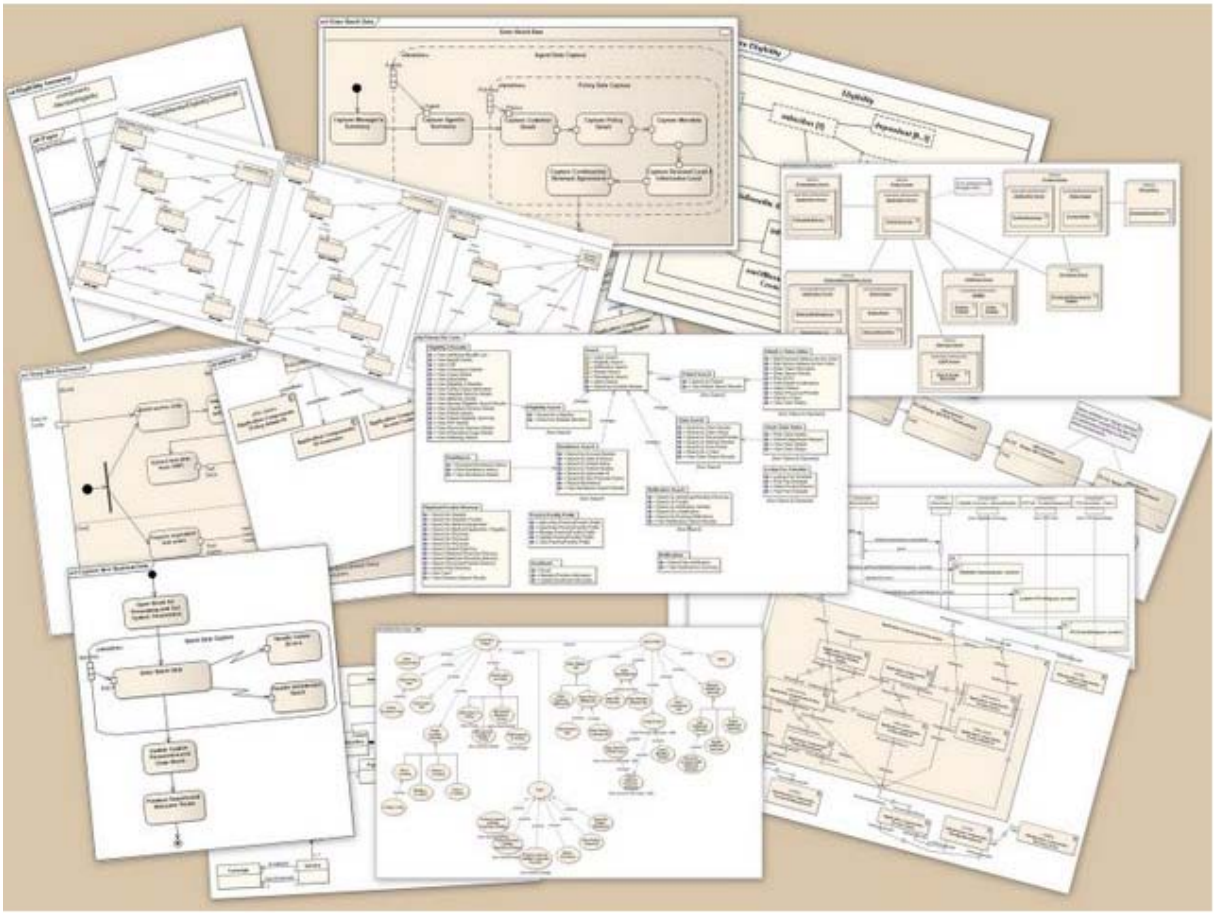
UML byl schválen OMG ™ (Object Management Group ™) jako standard v roce 1997. V dalších letech vzniká několik verzí modifikací modelovacího jazyka, které je v současné době shrnuto do poslední verze UML 2.3, která slouží jako sjednocující standard mezi analytiky, návrháři a programátory používajícími objektové technologie. V moderním procesu projektování informačních systémů má UML své pevné místo, jedná se o standard, nikoliv pouze o dočasnou pomíjivou záležitost.

Jaké důvody nás vedou k využití UML:

- Poskytnutí jednoduchého vizuálního modelovacího nástroje.
- Nezávislost na programovacím jazyku a vývojovém procesu.

- Podpora výměny již hotových modelovaných systémů.
- Jednotný nástroj pro vývoj a modelování systémů.

UML 2 definuje 13 základních typů diagramů, které se člení do dvou základních skupin, tj. *strukturní diagramy* a *diagramy chování a interakcí*.



Obrázek 1.1 – Kolekce UML diagramů pro projektovaný systém

Strukturní diagramy

Strukturní diagramy popisují statickou strukturu modelu. Jedná se o konkrétní objekty pro sestavení modelu, kterými jsou třídy, objekty, rozhraní a fyzické komponenty. Pro jejich vzájemné souvislosti se pak využívá popisných prvků pro závislosti a vztahy mezi jednotlivými objekty.

- Diagramy balíčků (Package Diagrams).
- Diagramy tříd (Class Diagrams).
- Strukturní diagramy (Structure Diagrams).
- Diagramy instancí (Objects Diagrams).
- Diagramy komponent (Component Diagrams).
- Diagramy nasazení (Deployment Diagrams).

Diagramy chování a interakcí

Diagramy chování zachycují stavy vzájemného působení a okamžitých vlivů uvnitř modelu tak, jak probíhají v daném čase, sledují, jak bude systém reagovat na podněty vnějšího okolí a charakterizují účinky operací nebo událostí, včetně jejich výsledků.

- Diagramy užití (Use Case Diagrams).
- Diagramy aktivit (Activity Diagrams).
- Stavové diagramy (State Diagrams).
- Diagramy komunikace (Communication Diagrams).
- Sekvenční diagramy (Sequence Diagrams).
- Diagramy časování (Timing Diagrams).
- Diagramy přehledu interakcí (Interaction Overview Diagrams).

Aplikace typů strukturních diagramů nebo diagramů chování a interakcí se využívá v různých etapách vývoje individuálně, zpravidla v rámci analýzy se řeší diagramy případů užití, scénáře činností, diagramy aktivit a diagramy tříd. V rámci návrhu modelovaného systému se z diagramů tříd mohou odvozovat datové modely, dále se upřesňují diagramy aktivit a z oblasti fyzického návrhu se analytik zabývá tvorbou diagramu komponent a diagramu nasazení. Poslední, tj. implementační fáze se zabývá diagramy tříd, z nichž se generuje programový kód, datové modely se realizují pro databáze a dále se dle diagramů komponent a diagramů nasazení realizují další podpůrné prvky zajišťující vztahy částí programových kódů v rámci různých platforem.

Softwarová podpora UML

Nástrojem pro vznik diagramů existuje v současné době řada softwarových produktů. Některé jsou určeny pouze ke grafické implementaci diagramů. Jejich vlastnosti jsou omezené v oblasti implementace a testování, pro něž nemá žádné podpůrné funkce. Profesionální nástroje jsou určeny k realizaci všech fází tvorby modelovaného systému, včetně jejich dokumentace s podporou šablon vestavěných nebo i vlastních, předem vytvořených.

Na závěr je uveden stručný přehled profesionálních produktů pro podporu UML: Rational Rose (licenční produkt), Violet UML Editor (bez licence), Borland Together 2008 (licenční produkt), Umbrello UML Modeller (GPL licence), Argo UML (BSD licence), Open ModelSphere (GPL licence), Visual Paradigm for UML (licenční produkt), Microsoft Visio (dostupný s MS Office), Enterprise Architect (licenční produkt).

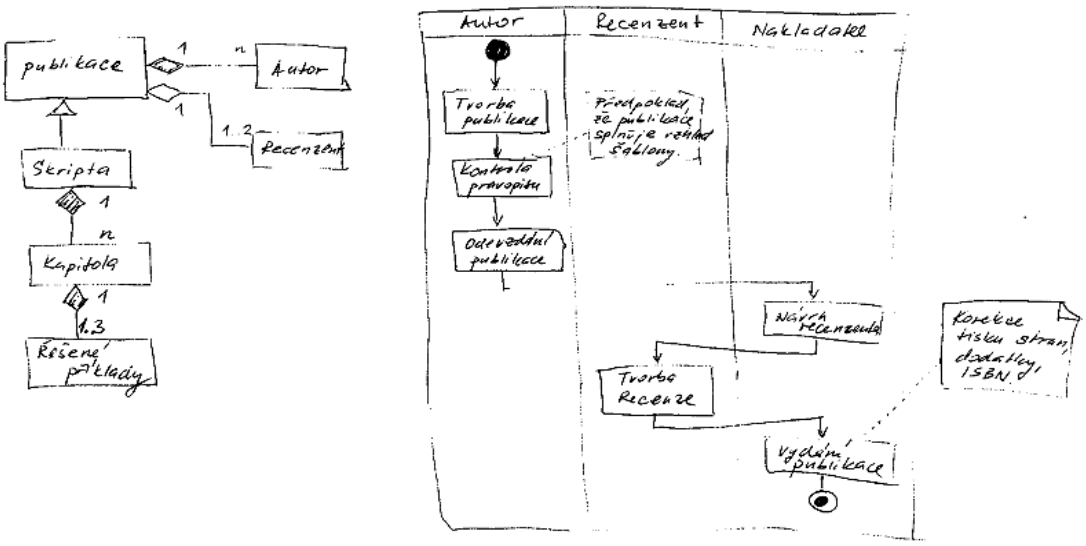
Ukázky v této publikaci budou zpracovány v programovém prostředí Enterprise Architect (oficiálního distributora Sparx Systems), který byl vybrán z hlediska poměrně velkého počtu nabízených funkcí, poměrně širokého spektra uživatelů a již vytvořených modelů a také z hlediska přiměřené ceny.

Způsoby použití UML

Oblast modelování je velmi široká a nabízí různé možnosti a hloubku jejího využití. Analytici nebo i ostatní lidé mohou využívat tohoto grafického jazyka na různé úrovni. Obecně se tedy přístup k modelování systémů rozděluje do čtyř úrovní: *kreslení konceptu, ukázka UML, kreslení detailních návrhů* a *UML jako programovací jazyk.*

Ať už je zvolena jakákoliv úroveň využití UML, vždy je obecně možné k modelování přistupovat dvěma směry, které se označují jako **dopředné** a **zpětné inženýrství**. Dopředné inženýrství se zabývá nejprve tvorbou analýzy a návrhu modelu IS a pak přistupuje k tvorbě programového kódu, naopak zpětné inženýrství přistupuje k tvorbě modelu IS z již existujícího programového kódu za účelem jeho dodatečného popisu nebo jeho dalšího rozšíření, které není vhodné dále již provádět bez podrobné analýzy v UML.

- **Kreslení konceptu** – při řešení problematiky objektů a jejich vztahů v jakékoliv řešené oblasti je možné využít nástrojů UML pro zakreslení analytického návrhu, což je vizuálně mnohem lepší metodika než dané vztahy popisovat slovně.
- **Ukázka UML** – použití jen části nástrojů UML pro komunikaci mezi analytiky pro zaznamenávání myšlenek a návrhů (obrázek 1.2). Cílem je srozumitelnost, rychlost analytického návrhu před implementací programu.



Obrázek 1.2 – Ukázka použití UML nástrojů verzi analýzy a návrhu

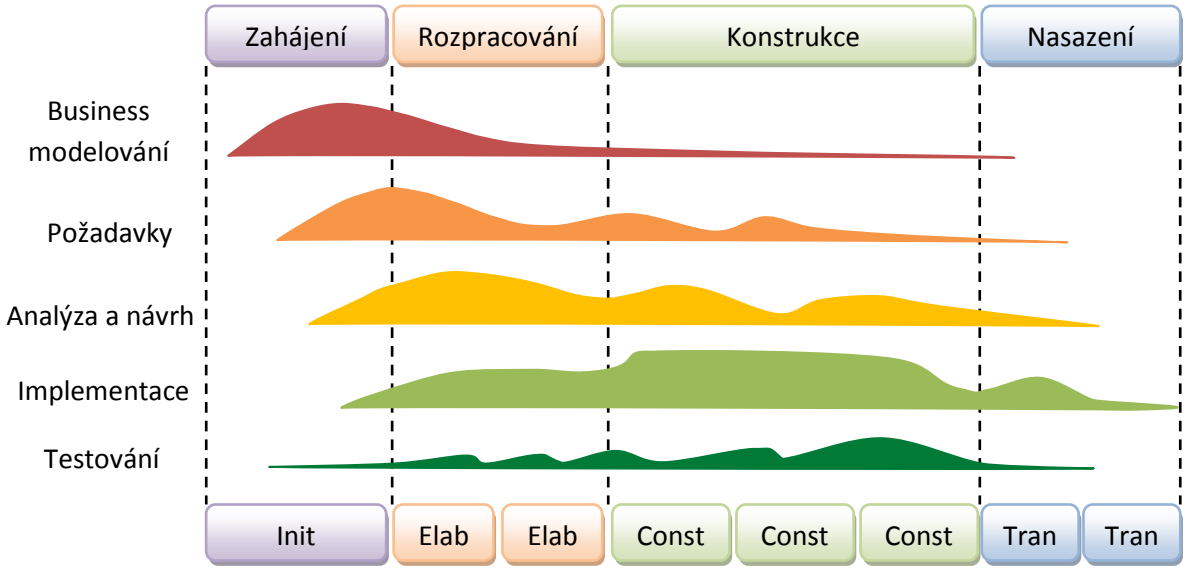
- **Kreslení detailních návrhů** – UML se využívá pro kompletní návrh a jeho realizaci. Důraz na podrobnou tvorbu analýzy tak, aby programátoři byli schopni rychle a snadno implementovat modelovaný systém bez potřeby dalších konzultací s uživatelem.
- **UML jako programovací jazyk** – analytik navrhuje UML diagramy, generuje spustitelný kód, proto jsou požadovány specializované nástroje analýzy a striktní vyjadřování v jednotlivých navržených diagramech. Dodržení standardu MDA (Model Driven Architecture) je nutné.

1.2 Unifikovaný proces

Unifikovaný proces (Unified Process, dále jen UP) je potřebný pro realizaci projektů a představuje proces vývoje softwaru v rámci životního cyklu (obrázek 1.3). Fáze vývoje jsou rozděleny do čtyř základních linií:

- **Zahájení** – období určené k plánování procesu, zachycení požadavků a identifikace možných rizik.
- **Rozpracování** – fáze zahrnuje návrhy a realizace procesu modelování, definice kvality a zpřesnění rizik a požadavků.
- **Konstrukce** – etapa počáteční tvorby a realizace modelovaného systému v daném softwarovém prostředí.
- **Nasazení** – představuje přechod navrženého software směrem k uživateli, pokud software dosáhl předpokládané úrovně a stability. Další chyby budou opraveny v rámci dalších verzí systému. Primárním cílem přechodné fáze je předat software uživateli, podpořit jeho schopnosti v obsluze a předat dokumentaci.

Fáze životního cyklu UP lze rozdělit podle velikosti modelovaného systému na více iterací (Elab #1, Elab #2, Const #1, Const #2, apod.). V těchto iteracích se pak realizují postupně pracovní postupy, kterých se daná fáze dotýká. Např. ve fázi zahájení bude většina času věnována Bussiness modelování, ostatní pracovní postupy se v této fázi podílí na vzniku modelovaného systému mnohem méně.



Obrázek 1.3 – Životní cyklus unifikovaného procesu

Pracovní postupy, které se podílí na jednotlivých fázích životního cyklu vývoje software:

- **Business modelování** – období určené k plánování procesu, zachycení požadavků a identifikace možných rizik.

- **Požadavky** – činnost sestavení funkčních a nefunkčních požadavků zahrnující činnost modelovaného systému a provázání požadavků s konkrétními činnostmi analýzy.
- **Analýza a návrh** – cílem je ukázat, jak bude systém realizován. Zahrnuje také objektovou analýzu, jejímž výsledkem jsou zpravidla diagramy tříd strukturované do balíčků.
- **Implementace** – návrh organizace zdrojového kódu do vrstev, implementace tříd a objektů, testovat vyvíjené komponenty.
- **Testování** – ověřovat spolupráci mezi objekty, ověřovat začlenění všech komponent software, zjistit, zda byly splněny všechny požadavky. Test se vyhodnocuje dle čtyř kritérií: spolehlivost, funkčnost, výkon aplikace, výkon systému.

Nelze tyto pracovní postupy zpracovávat postupně, ale v jednotlivých fázích vývoje softwaru se činnost dělí do všech pracovních postupů, jejichž časové nároky závisí na konkrétní fázi vývoje. Pokud implementace a testování ukazují na chybu v návrhové úrovni, pak je nutné se k tomu vrátit zpět.

1.3 Standardní sémantika UML

Standardní sémantiku UML je možné doplnit standardními omezeními, která umožňují různou interpretaci prvků.

- **Příznaky** (Tags) – umožňují přidat k prvku diagramu další informace ve formě dvojice. Syntaktický zápis je následující

  {název atributu = hodnota},

  Název atributu může být libovolný. Příznaky se zapisují do složených závorek a připojují se k názvu prvku diagramu. Často se aplikují v případě specifikace verze, autora nebo konkrétních implementačních poznámek.

- **Stereotypy** (Stereotypes) – umožňují klasifikovat prvky, zapisují se jako klíčová slova, např.

  <<include>>, <<extend>>, <<boundary>>.

  Význam jednotlivých klíčových slov zde uvedených, tj. „include“, „extend“ a „boundary“ je dán sémantikou jazyka UML a definován pro konkrétní část diagramu. V následujících kapitolách se s některými konkrétními stereotypy setkáte.

- **Omezení** (Constraints) – umožňují specifikovat omezení pro prvky diagramů a zapisují se do složených závorek, např. konkrétní omezení může být dáno tímto zápisem

  {počet_dat < 500}.

Příznaky, stereotypy a omezení slouží jako doplněk a rozlišení konkrétních modelovaných situací a v následujících částech modelovaného systému budou aplikovány a konkretizovány na dané podmínky a situace.

Shrnutí pojmů

Základním pojmem učebního textu je bezesporu pojem UML, což je standard mezi návrháři, programátory a analytiky. Představuje ve zkratce modelovací jazyk, který je k dispozici všem uživatelům a návrhářům systémů a je jazykově takřka bezbariérový.

Dále se studenti mohli seznámit s konkrétními názvy diagramů dvou základních skupin, tj. strukturních diagramů a diagramů chování, které v dalších kapitolách budou objasněny podrobně.

Neméně důležitými pojmy pro analýzu systému a jeho časový vývoj jsou pojmy dílčích fází vývoje: zahájení, rozpracování, konstrukce a nasazení. V těchto fázích probíhá vývoj modelovaného systému systematicky tak, aby bylo zamezeno případným dílčím chybám v návrhu systému průběžným testováním dílčích navržených částí dříve než na jeho úplném konci. Je možné využít pro dílčí části zpracování modelovaného systému odborníky, tj. analytiky, návrháře, programátory, jejichž spolupráce a komunikace nad analýzou vede bez zbytečných časových ztrát a zásadních chyb odhalených průběžným testováním ke vzniku požadovaného modelu/systému.

Otázky

1. UML je určeno pro konkrétní implementační prostředí?
2. Kolik základních fází má životní cyklus vývoje modelovaného systému?
3. Strukturní diagramy popisují dynamickou nebo statickou strukturu modelovaného systému?
4. Může uživatel využít UML pro systém, který již existuje? Z jakého důvodu a proč by mohl UML využít?
5. V které fázi životního cyklu věnujete největší úsilí pro implementaci modelovaného systému?

2 POŽADAVKY MODELOVANÉHO SYSTÉMU

Čas ke studiu: 2 hodiny

Cíl: Po prostudování tohoto odstavce budete umět

- Definovat první důležitou část návrhu modelovaného systému.
- Sestavovat funkční a nefunkční požadavky.
- Modelovat případy užití jako základní prvek návrhu modelovaného systému.
- Navrhovat základní a alternativní scénáře k dílčím případům užití
- Modelovat případy užití na pokročilé úrovni návrhu.

Výklad

Správa požadavků zahrnuje základní činnosti, které musí být při modelování systému splněny:

- Zajistit a zpracovat dokumentaci požadované funkčnosti a omezení systému.
- Vyhodnocení změn funkčních a nefunkčních požadavků a jejich důsledků.
- Evidovat dokumentaci jednotlivých rozhodnutí.

Požadavek je vlastnost nebo podmínka modelovaného systému. Takový systém pak musí splňovat nezbytné atributy kvality softwarového systému (nebo jinak realizovaného) z hlediska ***funkcionality***, ***použitelnosti***, ***spolehlivosti***, ***výkonnosti*** a ***podporovatelnosti***.

V následujících podkapitolách je uveden stručně výklad problematiky analýzy požadavků, uvedení doporučené syntaxe a uvedený příklad, který bude provázet i další kapitoly v této publikaci. Popis modelovaného systému, jako příkladu pro analýzu a modelování informačního systému, je uveden v kapitole 2.1.

Řešený příklad

2.1 Problematika modelovaného informačního systému

Podstatou modelovaného systému je sběr dat laboratorních úloh v klientském informačním systému (IS) a jejich následné zpracování a vyhodnocení ve formě laboratorního protokolu.

Laboratorní úlohy jsou v současné době koncipovány tak, že student ve většině případů pouze naměří data, jejichž výsledky zpracuje do výsledné podoby laboratorního protokolu bez nutnosti manuálního zásahu do zapojení hardwarových částí dané úlohy. Možnosti počítačových sítí v současné době nabízí také nekontaktní přístup k daným

laboratorním úlohám s využitím možností spouštění těchto úloh dálkovou správou zařízení a počítačů.

Jednotlivé úlohy vyžadují teoretickou přípravu a seznámení se s problematikou úlohy a podstaty měření. Ve většině případů v současné době je teoretická příprava koncipována popisem na webových stránkách laboratoře v textové podobě. Zpětná kontrola připravenosti studentů před zahájením měření dané úlohy chybí. Informační systém podpoří tuto kontrolu náhodným výběrem několika otázek v podobě testu jako vstupního rozhraní k části informačního systému s podporou měření, zpracování a vyhodnocení laboratorní úlohy.

Informační systém slouží také jako podpora zpracování dokumentace a výsledků měření využitím webových formulářů, které vyžadují vyplnění nezbytných obsahových částí, bez nichž by protokol nebyl přijat ke kontrole a hodnocení.

Informační systém musí splňovat řadu bezpečnostních prvků pro odstranění vlivu plagiátorství, které budou zpřesňovány ve fázi testování informačního systému. Rovněž zde budou zahrnuty zabezpečovací prvky plagiátorství při měření dat automatickým sledováním průběžného přihlašování do systému jednotlivými klienty a také ke kontrole shodnosti textů mezi jednotlivými výsledky klientů IS.

Pro splnění potřeb archivace studijních materiálů vždy na dobu tří let pak budou studenti automaticky generovat jednotlivé protokoly ve vhodném typu dokumentu, které budou sloužit k dokumentaci o jeho zpracování.

Tento počáteční popis problematiky není konečnou definicí všech potřeb informačního systému, ale je určen jako odrazový můstek pro první studii návrhu modelovaného systému. Další komunikace mezi analytikem a zákazníkem objektové analýzy budou v neustálém paralelním sledu s tvorbou modelovaného systému.

2.2 Analýza funkčních požadavků

Výklad

Představy zákazníka realizované analýzy a modelu informačního systému, které jsou částečně shrnuty v popisu problematiky modelovaného IS (kapitola 2.1), budou dále rozšířeny a konkretizovány v podobě funkčních požadavků. Nejprve bude sestaven seznam funkčních požadavků, které budou opatřeny vhodným identifikačním kódem tak, aby později bylo zřejmé přiřazení konkrétních diagramů modelovaného systému k danému funkčnímu požadavku. V našem případě modelovaného systému jsme zvolili kód čtyřmístný, kde první písmeno označuje typ požadavku, tj. FR (NFR), značí funkční (nefunkční) požadavky. Druhé až čtvrté místo v kódu je dáno pouze číslicemi, které označují pořadí uvedeného požadavku.

Funkční požadavky se řadí do čtyř základních skupin podle priority jejich naplnění. Označují se danou zkratkou na konec definice funkčního požadavku.

Syntaxe funkčních požadavků

<id> <systém> **bude** <funkce> priorita_požadavků

kde jednotlivé části syntaxe označují:

<id> - číslování funkčního požadavku,

<systém> - osoba, věc nebo událost vyvolávající funkci systému,

<funkce> - požadovaná činnost modelovaného systému.

Priorita požadavků v syntaxi funkčního požadavku je dána mírou nezbytnosti splnění funkce daného požadavku.

M – (must), požadavek musí být splněn,

S – (should), požadavek musí být splněn, pokud je to vůbec možné,

C – (could), požadavek může být splněn, pokud to nemá vliv na další požadavky,

W – (won't), požadavek nyní splněn nebude, ale v budoucím čase bude požadován.

Příklad syntaxe funkčního požadavku:

(chybně) FR01 Přihlášení klienta k informačnímu systému. M

(správně) FR01 Klient **bude** přihlášen k informačnímu systému. **M**

Řešený příklad

Funkční požadavky ukázkového modelovaného systému

FR01 Klient bude přihlášen k informačnímu systému. M

FR02 Správce systému bude evidovat klienty IS. M

FR03 Systém bude monitorovat klienty IS. S

FR04 Klient bude spravovat obsahové části laboratorních protokolů. M

FR05 Systém bude kontrolovat správnost plnění částí laboratorního protokolu. M

FR06 Systém bude evidovat výsledky klientů laboratorních úloh měření. S

FR07 Systém bude evidovat údaje o průběhu laboratorní úlohy měření. C

FR08 Správce systému bude evidovat teoretické přípravy laboratorních úloh měření. S

FR09 Správce systému bude evidovat vstupní testy. C

FR10 Klient bude řešit vstupní testy. C

FR11 Systém bude kontrolovat správnost vstupních testů a povolovat přístup k měření. C

FR12 Systém bude kontrolovat shodu textů mezi klienty IS. C

FR13 Klient bude generovat dokumentaci laboratorního měření. M

FR14 Klient bude požadovat správce systému o kontrolu laboratorního protokolu. M

FR15 Správce systému bude informovat klienty o výsledcích laboratorních protokolů. S

FR16 Klient bude požadovat správce systému o hodnocení pracovní činnosti (zápočet). S

FR17 Správce systému bude informovat o výsledcích pracovní činnosti (udělení zápočtu). S

FR18 Systém bude archivovat klienty IS. M

FR19 Systém bude archivovat laboratorní protokoly. M

Poznámka: Funkční požadavky byly sestaveny v pořadí jejich zjištění během komunikace se zadavatelem pracovní činnosti modelování informačního systému. Pro analytika by pak bylo vhodnější seřazení těchto funkčních požadavků dle priority.

CD-ROM

Analýza informačního systému je realizována v produktu Enterprise Architect, verze 8 firmy Sparx Systems Pty Ltd. První animační ukázka zahrnuje důležité části a postřehy při **sestavení funkčních požadavků**. Animace je zaznamenána programem Adobe Captivate 4.0.

2.3 Analýza nefunkčních požadavků

Výklad

Modelovaný systém požadované kvality musí splňovat požadavky, které se nevztahují k systémové funkcionalitě a představují pouze omezení nebo specifikaci dané implementace analyzovaného a modelovaného systému. Takové požadavky se označují jako **nefunkční**. Je možné vycházet z následujících obecných skupin nefunkčních požadavků:

- **Použitelnost** (zahrnuje přívětivost uživatelského prostředí, jednoduchost použití, dostupnost uživatelské dokumentace), např.
- **Spolehlivost** (schopnost chovat se správně v provozních podmínkách, měření frekvence a četnosti chyb při testování, předvídatelnost) , např.
- **Výkonnost** (přenosová rychlost, dosažitelnost, přesnost, doba odezvy, doba regenerace) , např.
- **Podporovatelnost** (udržovatelnost, testovatelnost a ostatní složky kvality požadované k implementaci systému připraveného pro jeho zařazení do provozu) , např.
- **Zabezpečení** (ošetření přístupu do systému neoprávněných osob, další bezpečnostní požadavky), např.

Syntaxe nefunkčních požadavků

<id> <předmět > **vyžaduje / vylučuje** <podmínka >

Příklady syntaxe nefunkčních požadavků:

NFR01 Koncový uživatel vyžaduje provedení objednávky do třiceti sekund.

NFR02 Koncový uživatel vyžaduje přístup k jakékoliv stránce do čtyř sekund.

NFR03 Systém vyžaduje splnění podmínky Service Level Agreement.

NFR04 Systém vyžaduje průměrnou dobu do poruchy minimálně čtyři měsíce.

NFR05 Všechny webové stránky vyžadují stahování do tří sekund při průměrné zatížení, a pěti sekund při špičkovém zatížení.

NFR06 Systém při vyhledávání vyžaduje schopnost zobrazování 500 výsledků vyhledávání na stránce.

NFR07 Systém vyžaduje možnost uživatelů tvořit nové pracovní postupy bez nutnosti dalšího programování.

NFR08 Systém vyžaduje možnost tvorby a plnění daňových tabulek pro nadcházející daňové období správcem systému.

NFR09 Systém vyžaduje k ověřování uživatele použití firemního jednotného systému Signon.

NFR10 Systém vyžaduje zabezpečený přístup k informacím o mzdách pouze danému správci.

Řešený příklad

Nefunkční požadavky ukázkového modelovaného systému

NFR01 Informační systém bude implementován v prostředí ASP.NET.

NFR02 Data budou uložena v databázovém serveru SQL, verze 2005 (z důvodu kompatibility s prostředím Matlab/.NET).

NFR03 Odevzdání protokolu se řeší zasláním automatické e-mailové zprávy s využitím prostředí MS Outlook 2007.

NFR04 Klienti vyžadují přihlášení k IS s využitím LDAP.

NFR05 Klienti vyžadují přístup z jakékoliv webové stránky do 4 sekund.

NFR06 Řešení vstupních testů vyžaduje časový limit maximálně 5 minut/klienta.

NFR07 Systém umožní správci systému aktualizovat vstupní testy.

NFR08 Systém umožní archivovat informace o klientech po dobu 5 let s maximálním počtem 20 klientů/rok.

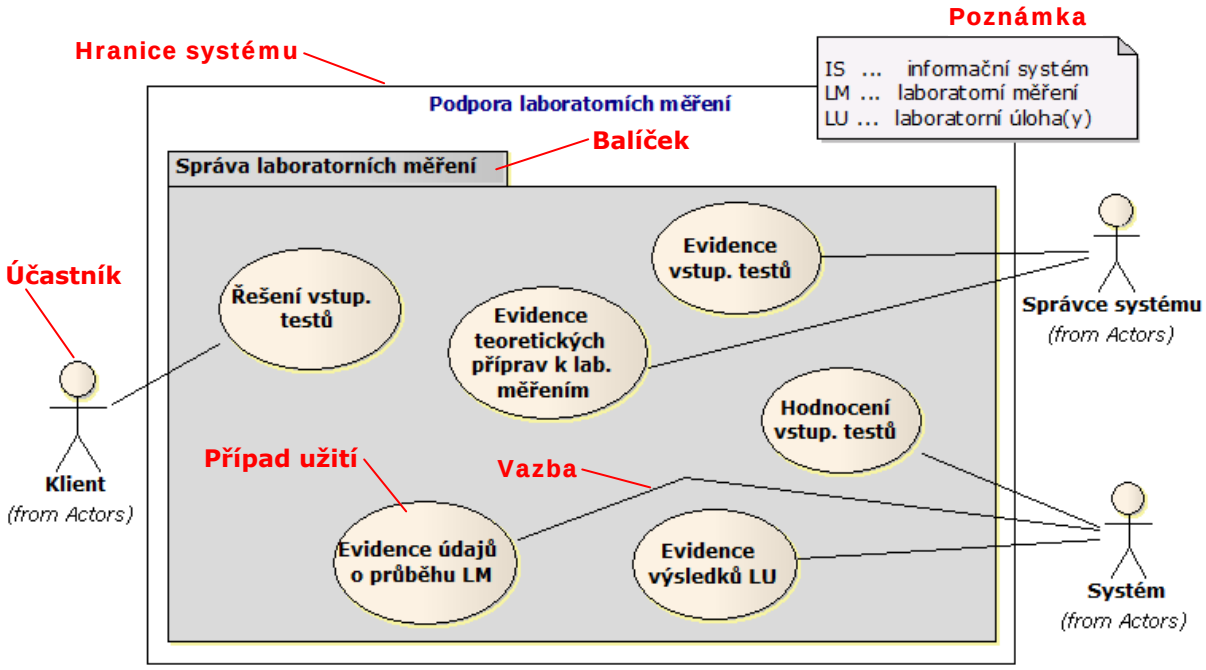
NFR09 Systém umožní archivovat laboratorní protokoly klientů po dobu 5 let s maximálním počtem 120 protokoly/rok.

Výklad

2.4 Modelování případů užití

V předchozí části byly představeny funkční a nefunkční požadavky, které jsou jednou z tradičních forem inženýrství požadavků. Podrobnější formou inženýrství požadavků je oblast **modelování případů užití**, které rozšiřují informace o modelovaném systému tím, že se uskuteční:

- definice hranic modelovaného systému,
- analýza účastníků,
- analýza případů užití,
- logické uspořádání případů užití,
- tvorba scénářů (doplňující informace o přesném postupu případu užití).



Obrázek 2.1 – Modelování případů užití v programu Enterprise Architect

Definice hranic modelovaného systému

Hranice modelovaného systému se vizuálně zobrazují obdélníkem, kde uvnitř jsou dílčí případy užití a vně jsou účastníci (obrázek 2.1). Ve skutečnosti však hranice systému představují souhrn požadavků, které má modelovaný systém splňovat. Zde je nutné podotknout, že někdy může nevhodná definice hranic systému vyvolat konflikt mezi zadavatelem a analytikem a to tehdy, pokud hranice systému nezahrnují vše, co si zadavatel představoval.

Analýza účastníků

Modelovaný systém má splnit požadavky, jejichž činnost se zpravidla aktivuje zvnějšku (za hranicemi systému) osobou nebo předmětem souhrnně nazývaným

účastník (Actors) daného modelu případů užití. Účastník je role, kterou má uživatel ve vztahu k systému. Účastníkem může být člověk nebo stroj, ale také další systém nebo subsystém modelu. V jazyce UML je účastník označován symbolem kreslené figurky včetně jejího označení umístěného pod ní (obrázek 2.1).

Při modelování situace v systému je možné, že konkrétní činnost je aktivována nikoliv osobou nebo systémem, ale konkrétním časovým údajem, ve kterém se tato činnost provede. Pak je vhodné zavést abstraktního účastníka s názvem Čas.

Analýza případů užití

Případ užití (Use Case) definuje posloupnost činností, které systém, podsystém nebo třída může vykonat prostřednictvím jejich aktivace účastníkem modelovaného systému. Symbolicky se případ užití obrazuje elipsou, která uvnitř obsahuje stručný popis daného případu užití (obrázek 2.1).

Logické uspořádání případů užití

Rozsáhlé modelované systémy je vhodné uspořádat do logických celků, tzv. **balíčků** (Package). Analýzu systému lze distribuovat ke zpracování dílčích částí se specifickým názvem odpovídající dané problematice. Existence jednoho balíčku v diagramu případu užití nemá význam, viz obrázek 2.1, který je ukázkou prvků diagramu případu užití.

Tvorba scénářů

Scénáře jsou nezbytnou součástí diagramů případů užití, zejména pro rozsáhlejší činnosti, kdy název případu užití nedefinuje plně cíl, postup a konkrétní komunikaci mezi dalšími účastníky. Sekvenční diagramy jsou v podstatě scénáře v grafické podobě zahrnující navíc vizualizaci následnosti jednotlivých kroků, podíl konkrétních tříd a instancí na dané činnosti případu užití, vznik nebo zánik konkrétních objektů apod. Pro jednoduché scénáře se nedoporučuje zbytečná tvorba sekvenčních diagramů, která by způsobila navýšení počtu diagramů v dané analýze.

Scénář případu užití specifikuje všechny navržené případy užití tím, že podrobně definuje:

- stav systému před spuštěním,
- konkrétní postup událostí po aktivaci daného případu užití účastníkem,
- stav systému po ukončení případu užití.

Syntaxe scénáře není pevně definovaná standardem UML, ale v podstatě ve všech publikacích jsou uvedeny scénáře ve formě seznamu (tabulky) viz tab. 2.1. Obecnou strukturu základního scénáře je vhodné sestavovat ve formě číslovaného seznamu.

Identifikátor jednoznačně určuje daný případ užití. Nijak číslování případů užití nenavazuje na číslování požadavků, protože jejich počet není stejný. Jeden funkční požadavek může být realizován jedním nebo i více případy užití a naopak.

Pokud nenastane chyba, která naruší postup činnosti případu užití, pak se provádí konkrétní případ užití podle základního scénáře. Analýza musí být řádně provedena také pro

nepředvídané situace, které mohou ovlivnit činnost modelovaného systému. Těm jsou určeny alternativní scénáře. Počet alternativních scénářů pro jeden případ užití může být 0, 1 nebo také více. Opět to závisí na konkrétním případu užití. Alternativní scénáře se označují názvem, který vystihuje nestandardní průběh případu užití nebo konkrétní chybovou situaci.

Základní a alternativní scénář je formátován v podobě jednoúrovňového nebo víceúrovňového seznamu. Řízení toku činnosti případů užití, obdobně jako v programování, se označuje klíčovými slovy, která označují typ řízení činnosti.

Podmíněné kroky postupu činnosti jsou vyjádřeny klíčovým slovem KDYŽ (+ JINAK). Opakování kroků činnosti se vyjadřuje klíčovým slovem PRO, za nímž následuje upřesnění počtu opakování, např. „PRO každou šestou úlohu laboratorního měření“. Pokud opakování kroků činnosti není přesně stanoveno na daný počet opakování a je závislé na splnění (nesplnění) konkrétní podmínky, pak se využívá klíčové slovo DOKUD. Všechny tyto zmíněné postupy, které se obecně využívají k řízení toku činnosti, jsou ve scénáři odlišeny podřízenou úrovní číslovaného seznamu pro všechny jeho dílčí kroky (tab. 2.1).

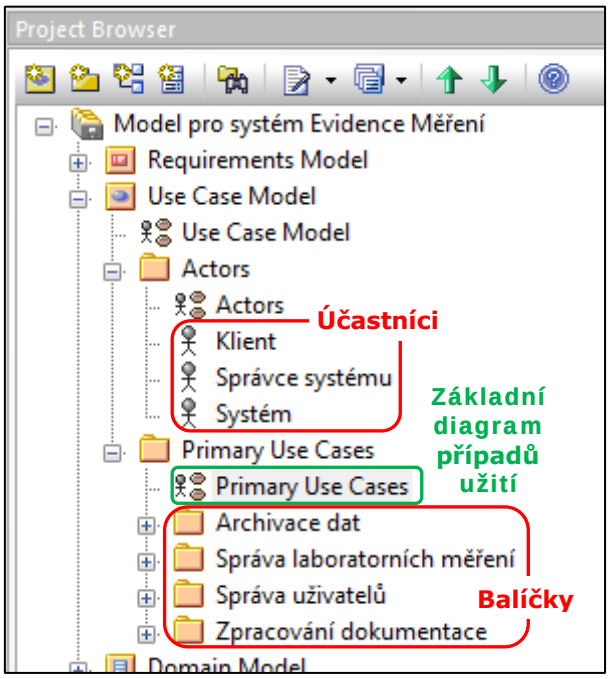
Tab. 2.1 – Formální vzhled scénáře případu užití.

Název případu užití	**Přihlášení k IS**
Identifikátor	UC01
Cíl	Zpřístupnění činnosti informačního systému.
Primární účastník	Uživatel systému
Sekundární účastník	Systém, Správce systému
Vstupní podmínky	IS je v daném období (semestru) přístupný.
Výstupní podmínky	Uživatel je přihlášen k IS.
Základní scénář	1. *Uživatel systému* zadá požadavek na přihlášení. 2. Systém umožní přístup k vyplnění přihlašovacích údajů. 3. DOKUD *Uživatel systému* trvá na požadavku přihlášení k IS, pak: 3.1. *Uživatel systému* zadává přihlašovací údaje. 3.2. *Uživatel systému* požaduje přihlášení k IS. 3.3. Systém ověřuje zadané údaje pro přihlášení. 3.4. KDYŽ jsou údaje pro přihlášení správné, PAK: 3.4.1. *Uživatel systému* je přihlášen k IS. 3.5. JINAK: 3.5.1. Systém otevírá opět formulář pro změnu přihlašovacích údajů.

Alternativní scénář(e)	**Překročený limit možností přihlášení k IS** (spustí se v kroku 3.3., když je překročen limit možných přihlášení k IS) 3.3.1. DOKUD *Uživatel systému* požaduje aktivaci přístupu k IS, pak: 3.3.1.1. *Systém* požaduje kontaktní e-mail pro zaslání informace o aktivaci přístupu k IS. 3.3.1.2. *Uživatel systému* zasílá žádost o aktivaci přístupu k IS *Správci systému*. 3.3.2. KDYŽ *Uživatel systému* požaduje změnu hesla 3.3.2.1. Rozšíření **UC03 Změna hesla**. 3.3.3. JINAK Uživatel systému požaduje ukončení činnosti IS.

Syntaxe zápisu scénářů v této publikaci odlišuje názvy případů užití **zvýrazněním textu tučně**. Klíčová slova vyjadřující tok řízení činnosti se vyznačují VELKÝMI PÍSMENY. Účastníci, kteří se podílí na činnosti ve scénáři se zvýrazní *Textem se skloněným písmem*, jehož počáteční znak je zpravidla formátován velkým písmenem.

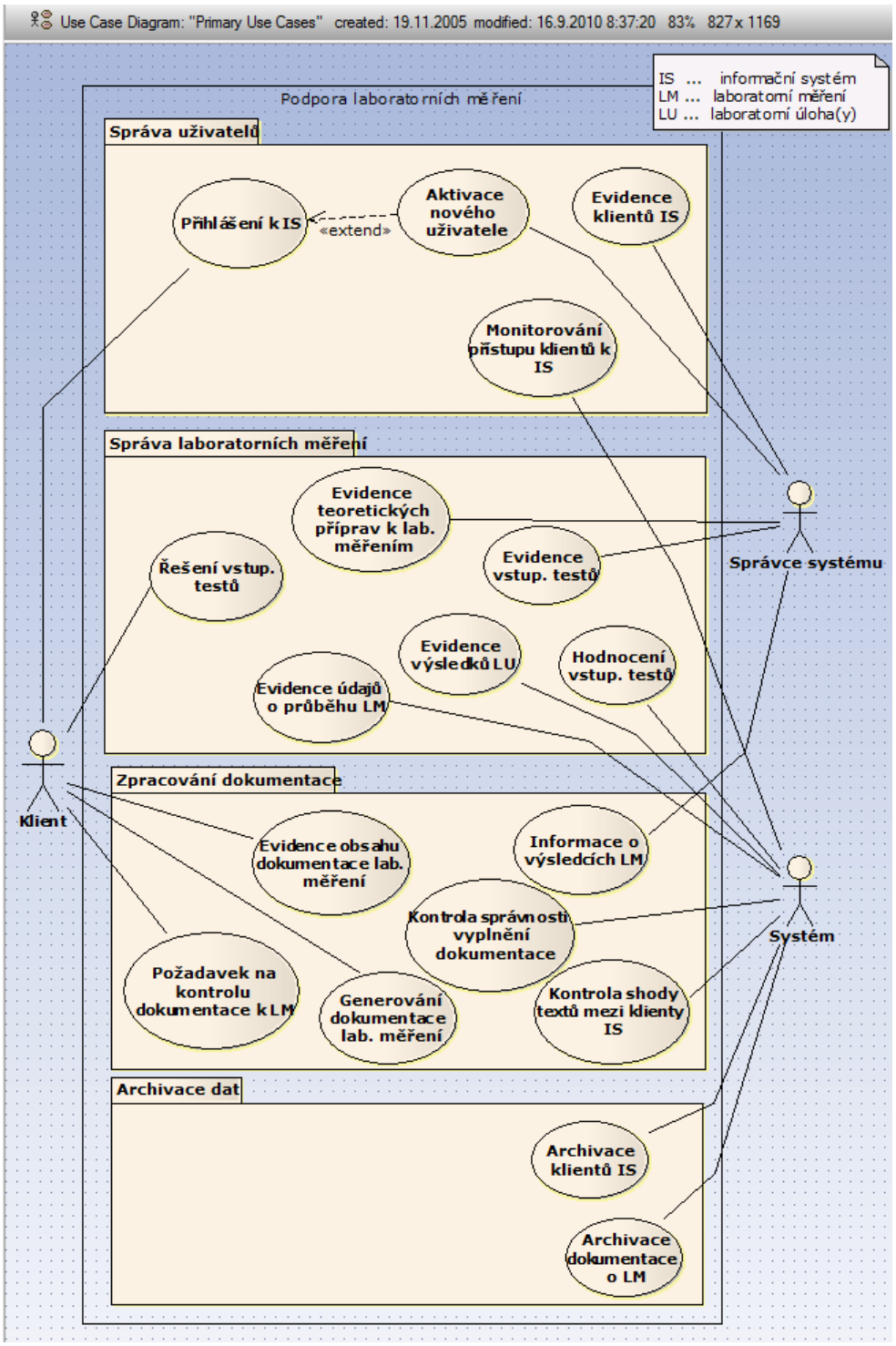
Je nutné definovat u alternativních scénářů úroveň kroku, kdy se spustí tato alternativní činnost. Pokud je alternativních scénářů více, pak jsou definovány následně za sebou podle pořadí úrovní, kdy jsou prováděny. Jsou uváděny v tabulce za sebou a vizuálně se oddělují tučně zvýrazněným názvem alternativního scénáře.



Obrázek 2.2 – Zobrazení okna Project Browser – ukázka členění informačního systému do čtyř balíčků, do nichž jsou zahrnuty příslušné případy užití

Řešený příklad

Modelovaný systém, jehož problematika je uvedena v kapitole 2.1, byl v první fázi návrhu rozčleněn na čtyři balíčky: Správa uživatelů, Správa laboratorních měření, Zpracování dokumentace a Archivace dat (obrázek 2.2).

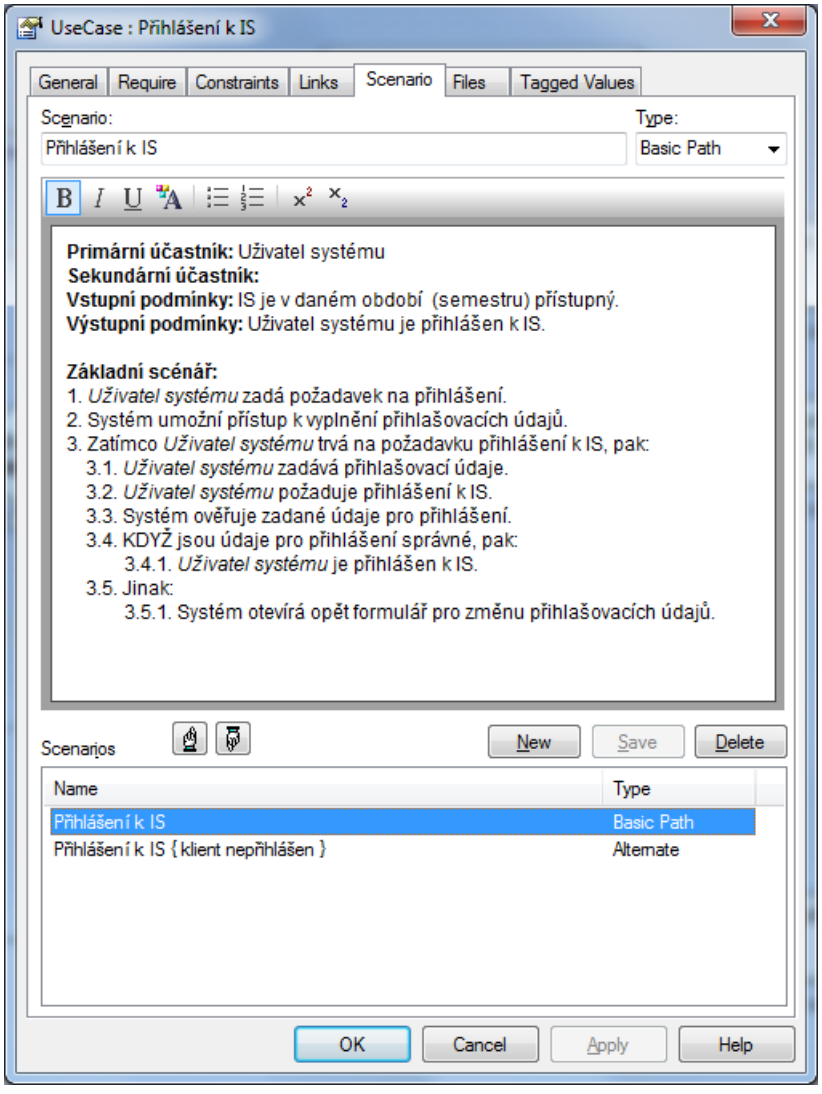


Obrázek 2.3 – Diagram případů užití (v obecné fázi návrhu) sestavený v programu Enterprise Architect

Informační systém je aktivován třemi účastníky, ať už se jedná o primární působení na systém (účastník Klient a Správce systému), tak i sekundární působení, které vychází z potřeb a vyššího komfortu služeb informačního systému.

Po sestavení hranic systému, balíčků a účastníků následuje analýza činností podle seznamu požadavků uvedených v kapitole 2.2. Do jednotlivých balíčků se vkládají případy užití, respektive jen jejich výstižné názvy. V této části kapitoly byl v první fázi proveden celkový návrh diagramu případu užití (obrázek 2.3).

Dále se provádí podrobný návrh scénářů jednotlivých případů užití (obrázek 2.4, obrázek 2.5). Až při této činnosti analytik často provádí změny struktury diagramu případů užití. Zde se vyjasňují vztahy mezi jednotlivými případy užití. Probíhá upřesňování a někdy dochází i k radikálním zásahům, např. rozpadu nebo slučování činností, které vyplývají z postupně sestavujících scénářů případů užití.



Obrázek 2.4 – Základní scénář pro UC01 Přihlášení k IS sestavený v programu Enterprise Architect

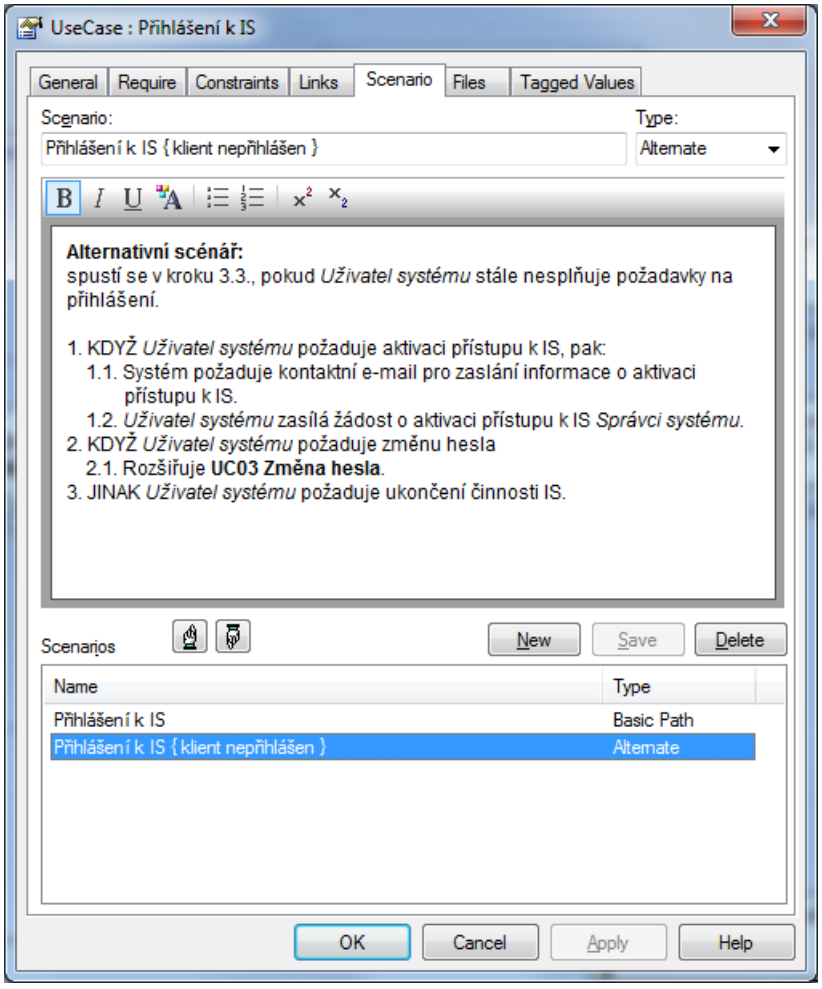
V další kapitole budou vyjasněny postupy pro pokročilé modelování případů užití, proto také i v řešených příkladech budou posléze ukázány další změny v analýze diagramu případu užití pro navrhovaný typ modelovaného systému.

Model případů užití a jejich specifikace musí být pro všechny účastníky tvorby informačního systému snadno čitelný, rozšiřitelný a jednoznačný. Proto i tvorba scénářů nepředpokládá, že by postup činností pro konkrétní případ užití měl zahrnovat činnosti i pro situace, které nastanou zřídka (chyba systému, nepředvídané přerušení apod.). Je tedy vhodné u případů užití specifikovat tzv. **základní scénář**, v němž je uveden běžný seznam činností, a ostatní činnosti pro neočekávanou situaci definovat v tzv. **alternativní scénář(e)**.

Identifikátory pro konkrétní modelovaný systém byly označeny syntaxí:

UC*n nazev*

kde *n* je pořadové číslo případu užití zvolené analytikem a *nazev* je výstižný a stručný název případu užití. Vzhledem k rozsahu modelovaného systému bude postačovat dvoumístné číslování případů užití, tj. od UC01 až UC99.



Obrázek 2.5 – Alternativní scénář pro UC01 Přihlášení k IS sestavený v programu Enterprise Architect

CD-ROM

Analýza informačního systému je realizována v produktu Enterprise Architect. Animační ukázky zahrnují důležité části a postřehy při **sestavení případů užití** a **tvorby scénářů**. Animace jsou zaznamenány programem Adobe Captivate 4.0.

Výklad

2.5 Pokročilé modelování případů užití

U složitějších diagramů případů užití je snaha o zjednodušení, lepší srozumitelnost diagramu případu užití. Toho je možné dosáhnout *zobecněním účastníků* nebo *zobecněním případů užití*, což vizuálně představuje snížení počtu grafických prvků v diagramu případů užití, ale co je v analýze podstatnější, že modelovaný systém bude realizován na principech objektového modelování.

Pro objasnění a zpřesnění jednotlivých vztahů mezi případy užití a účastníky nebo případy užití navzájem se aplikují nové typy vazeb či závislostí, které se definují značením *klauzulí* <<*include*>> nebo <<*extend*>>.

Zobecnění

Vztahu zobecnění se využívá v diagramech z důvodu celkového zjednodušení neboli snížení počtu grafických prvků tam, kde analytik vidí závislost jednoho případu užití k různým účastníkům. Cílem zobecnění je tedy získat přehledný diagram a udržovatelnost diagramů případů užití. V implementačním prostředí programovacích jazyků pak zobecnění představuje aplikaci objektově orientovaných principů.

Použít lze *vztah zobecnění* jak pro účastníky, pak se jedná o *zobecnění účastníků*, tak i u případů užití, tj. *zobecnění případů užití*.

Zobecnění účastníků

Z diagramu případů užití je patrné, že více účastníků provádí několik totožných případů užití, protože ve skutečnosti se rozsah činností může překrývat. Proto se zavádí rodičovský účastník, který provádí většinu případů užití. Vazby na případy užití od jeho potomků jsou jen tehdy, když potomci provádí specifické činnosti.

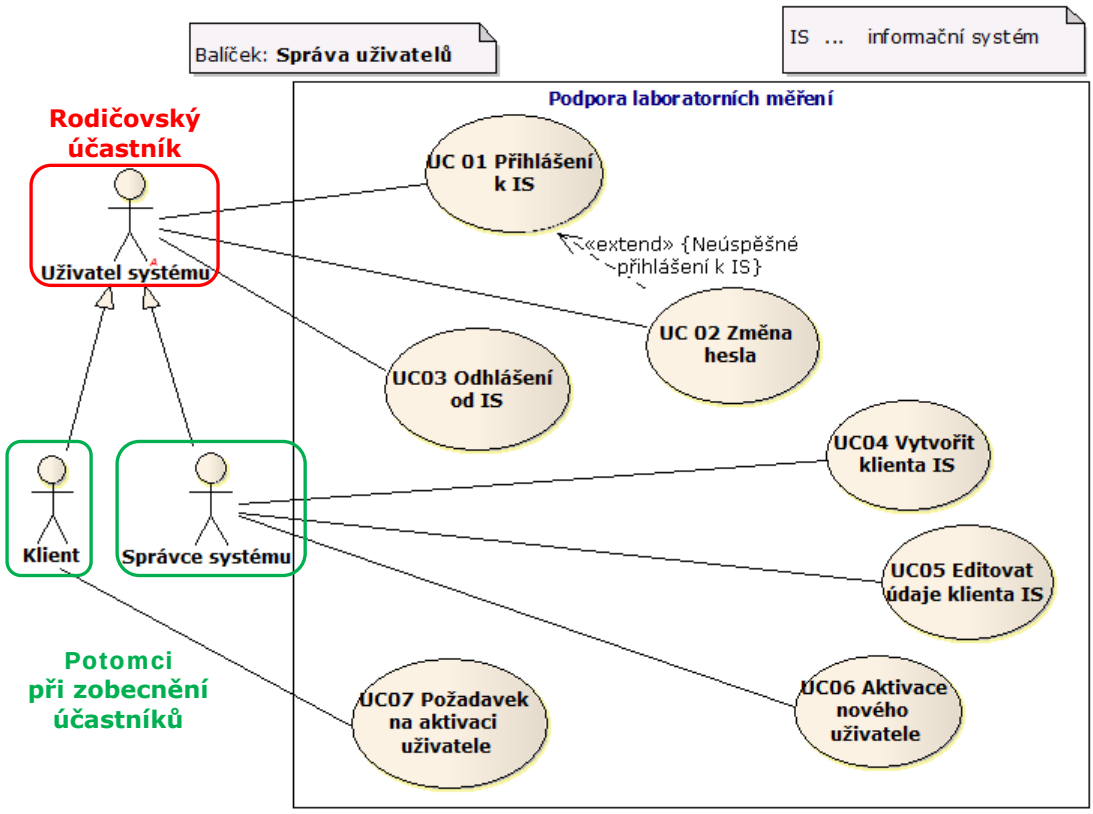
Pokud rodičovský účastník v reálném systému existuje, pak se mezi tímto rodičovským účastníkem a potomkem zavádí vazba zobecnění. Pokud rodičovský účastník v reálném systému neexistuje, pak se zavádí do systému *abstraktní účastník*, který má pro modelovaný systém jen účelovou existenci ke zjednodušení modelu systému.

Od rodičovského účastníka nebo i jeho potomků směřují k jednotlivým případům užití vazby dle potřeby. Vztah zobecnění se označuje šipkou se zakončením ve tvaru prázdného trojúhelníku směrem k rodičovskému účastníku.

Pro jeden totožný případ užití od více osob je spíše neúčelné zavádět abstraktního účastníka. Diagramy případy užití by měly směřovat k jednoduchosti.

Řešený příklad

V balíčku **Správa uživatelů** existují případy užití společné pro účastníka **Klient** a účastníka **Správce systému**. Pro zjednodušení diagramu případů užití je vhodnější provést zobecnění účastníka zavedením abstraktního účastníka **Uživatel systému**. Tímto se diagram případů užití stává přehlednějším pro členění případů užití než v původním případě, kdy by k daným případům užití přistupovaly vazby od několika účastníků. V dalších balíčcích má **Klient** další vazby na jiné případy užití. Zde je pouze část modelovaného systému.



Obrázek 2.6 – Zobecnění účastníků

Výklad

Zobecnění případů užití

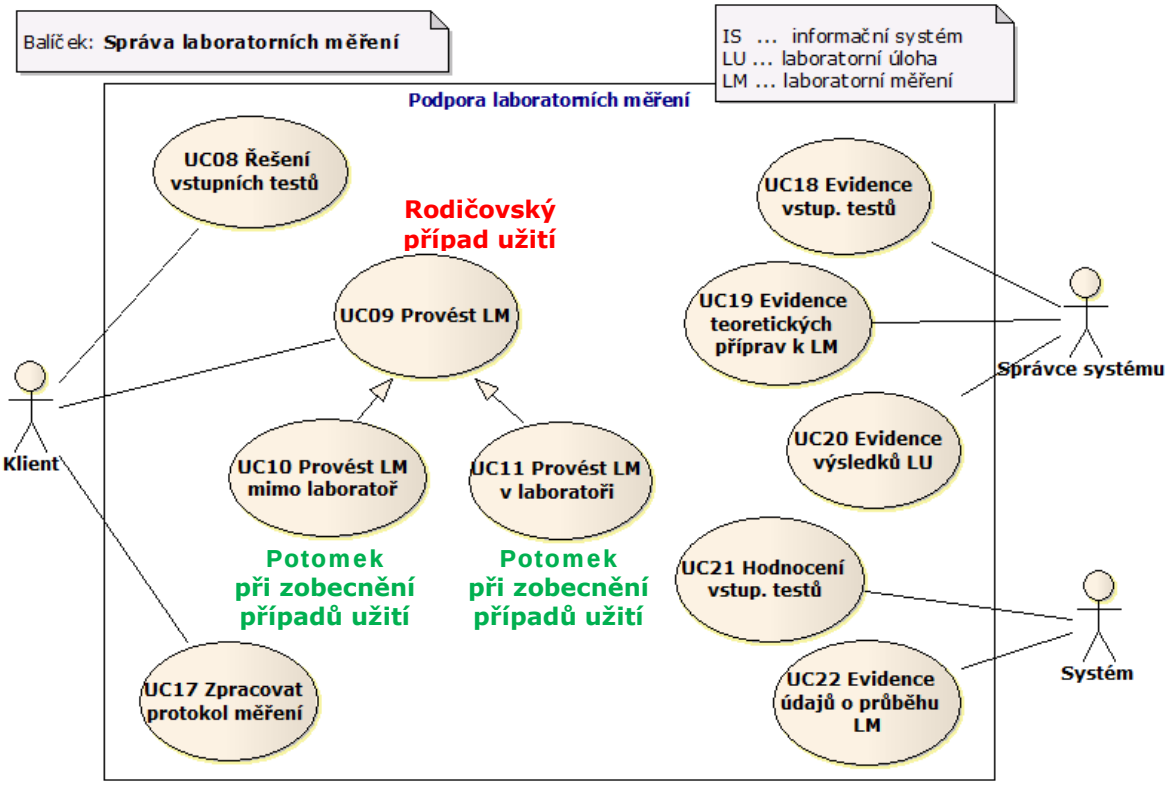
Případ užití může mít také zobecněný tvar případu užití, jehož funkce a vlastnosti mohou být děděny do úrovní potomků (případ užití specializovaný pro konkrétní případ, který navazuje zobecněnou závislostí směrem k zobecněnému případu užití). Je nutné při implementaci zobecněných závislostí na rodičovský případ užití uvažovat, zda tato vazba je určena správným typem závislosti. Je možné, že pro nezkušené analytiky by mohlo dojít k záměně se závislostí typu <<extend>>. Rozdíl je v těchto závislostech v tom, zda případy užití mají společné vlastnosti či metody nebo jsou odlišné, ale svou činností navazují na stejný případ užití.

Obdobně jako u zobecnění účastníků, také pro zobecnění případů užití dvou potomků může vzniknout *abstraktní případ užití*, který má společné atributy nebo operace, ale v reálném systému takový rodičovský případ užití samostatně nebude volán žádným účastníkem. Vazba zobecněných případů užití je značena šipkou stejného tvaru jako u zobecnění účastníka se směrem k rodičovskému případu užití.

Řešený příklad

V balíčku **Správa laboratorních měření** byl analyzován vznik zobecněných případů užití pro rodičovský případ užití **Provést LM**. Při analýze problematiky způsobu měření laboratorních úloh existují dvě varianty činností. Jedna skupina laboratorních úloh je plně podporovaná možností měření s podporou internetu nebo vzdálené plochy a systému IP kamer pro audiovizuální přenos. Druhá skupina laboratorních úloh je určena výhradně pro měření v laboratoři s nutností fyzického zásahu nebo tvorby částí laboratorních úloh. Obecný postup daný scénáři obou skupin je podobný, proto je možné definovat rodičovský případ užití **Provést LM**, který je zobecněním případu užití **Provést LM mimo laboratoř** a **Provést LM v laboratoři**.

Důležité je ověření oprávněného použití zobecnění případu užití tím, že budou sestaveny scénáře pro potomky rodičovského případu užití a posouzením podílu společných kroků scénářů rodičovského případu užití a jeho potomků.



Obrázek 2.7 – Zobecnění případů užití

Scénáře rodičovských případů užití a případy užití potomků jsou typické využitím společných činností, které jsou uvedeny v rodičovském případu užití a specifické činnosti jsou

dále konkretizovány v případech užití potomků. Také je doporučeno do scénáře potomka uvést návaznost na rodičovský případ užití bezprostředně za názvem případu užití potomka.

Ve scénářích nebudou dále uvedeny možné alternativní scénáře, protože seznam všech tří scénářů by byl podstatně delší. Alternativních scénářů pro reálný provoz a měření dat by bylo zajisté více z důvodu komplikací spojení přes vzdálenou plochu, nezbytnosti zúčastněných provozních míst apod. Hlavním cílem je v tomto příkladu ukázat sestavení scénáře v základní části a provázanosti scénářů rodičovského (abstraktního) případu užití a jeho potomků (specifikací), které jsou označeny v základním scénáři specifikačním bodem, např. **{Specifikace provedení LM}**. Základní scénář obsahuje společné činnosti a specifikační činnosti, které mohou být uskutečněny v jednom nebo i několika specifikačních případech užití, jsou uvedeny pouze ve scénářích potomků (specifikací).

Tab. 2.2 – Rodičovský scénář případu užití.

Název případu užití	**Provést LM**
Identifikátor	UC09
Cíl	Nové poznatky, informace a data z LM.
Primární účastník	Klient
Sekundární účastník	Správce systému, Systém
Vstupní podmínky	Pro vybranou laboratorní úlohu *Klient* splnil vstupní test.
Výstupní podmínky	Dostupné soubory dat z LM.
Základní scénář	1. Zahrnout **UC12 Výběr LU**. 2. Systém uchová datum a čas zahájení LM. 3. Systém spustí činnost videokamery a nahrávání činnosti LM. 4. KDYŽ LU nebyla realizována (měřena), pak: 4.1. Zahrnout UC13 Zobrazit pokyny k měření. **{Specifikace provedení LM}** 5. Systém zaznamená datum a čas ukončení měření. 6. Systém ukončí videozáznam LM. 7. Systém uloží soubor videozáznamu s generovaným názvem (klient, úloha, datum, čas) do výchozího adresáře. 8. Systém informuje *Klienta* o názvu souboru, kde byl uložen videozáznam. 9. Systém zaznamená, že bylo již provedeno měření vybrané LU. **{Specifikace provoz laboratoře}**
Alternativní scénář(e)	

Tab. 2.3 – Scénář případu užití potomka, konkrétně pro UC10.

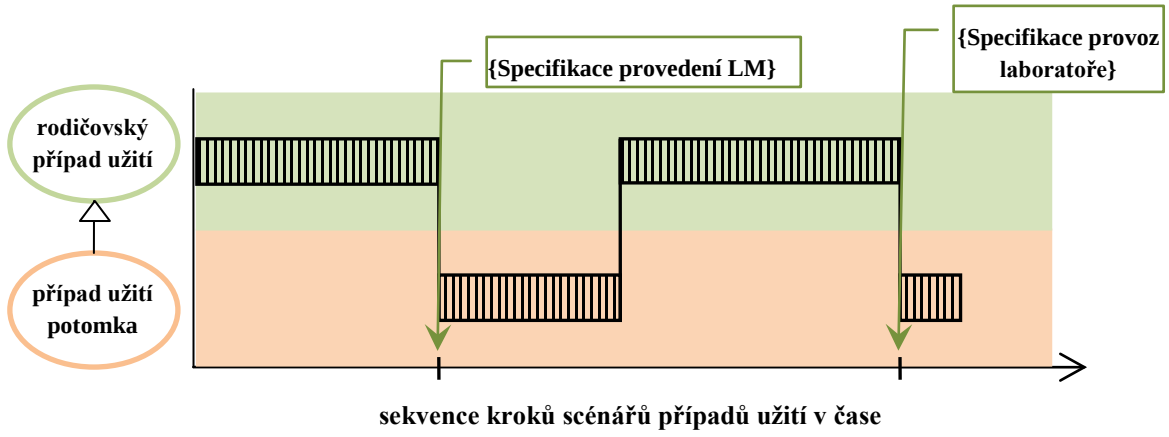
Název případu užití	**Provést LM mimo laboratoř (specifický UC)**
Identifikátor	UC10
Cíl	Nové poznatky, informace a data z LM.
Primární účastník	Klient
Sekundární účastník	Správce systému, Systém
Vstupní podmínky	Pro vybranou laboratorní úlohu *Klient* splnil vstupní test.
Výstupní podmínky	Dostupné soubory dat z LM.
Základní scénář	**v bodě {Specifikace provedení LM}** 1. Systém upozorní *Klienta*, že část aplikací musí uzavřít standardním způsobem uzavření aplikací Windows a vrátit se zpět k původnímu IS. 2. Systém spustí aplikaci prohlížeče s parametrem URL adresy pro vybranou LU. 3. KDYŽ je možné LU měřit pouze s využitím přístup ke vzdálené ploše, pak 3.1. Systém zobrazí informace o dílčích krocích k zpřístupnění LU (IP adresa, název účtu a heslo na vyžádání Správce systému, název aplikace na ploše vzdáleného počítače). 3.2. *Klient* spustí aplikaci na vzdálené ploše. 3.3. Zahrnout **UC14 Naměřit data**. 3.4. Zahrnout **UC15 Zálohovat data**. 3.5. *Klient* ukončí aplikaci standardním způsobem Windows. 3.6. *Klient* ukončí přístup k vzdálené ploše počítače. 4. Jinak: 4.1. Systém spustí IE s parametrem URL adresy podle vybrané LU. 4.2. Zahrnout **UC14 Naměřit data**. 4.3. Zahrnout **UC15 Zálohovat data**. 4.4. *Klient* ukončí IE standardním uzavřením Windows aplikace. 5. *Klient* předá informaci systému o ukončení činnosti v jiných aplikacích.
Alternativní scénář(e)	

Specifikace s označením {Specifikace provoz laboratoře} je součástí pouze scénáře případu užití **Provést LM v laboratoři**.

Tab. 2.4 – Scénář případu užití potomka, konkrétně pro UC11.

Název případu užití	**Provést LM v laboratoři (specifický UC)**
Identifikátor	UC11
Cíl	Nové poznatky, informace a data z LM.
Primární účastník	Klient
Sekundární účastník	Správce systému, Systém
Vstupní podmínky	Pro vybranou laboratorní úlohu *Klient* splnil vstupní test.
Výstupní podmínky	Dostupné soubory dat z LM.
Základní scénář	**v bodě {Specifikace provedení LM}** 1. Systém upozorní Klienta, že fyzické zapojení (realizaci) LU provede podle pokynů k LM. 2. Zahrnout **UC14 Naměřit data**. 3. KDYŽ LU vyžaduje uložení dat, PAK: 3.1. Zahrnout UC15 Zálohovat data. **v bodě {Specifikace provoz laboratoře}** 1. Systém upozorní studenta na ukončení činnosti fyzických zařízení a jejich přemístění na původní místo v laboratoři.
Alternativní scénář(e)	

Objasnění časové posloupnosti kroků a toku řízení mezi zobecněnými případy užití, kdy účastník požaduje vykonání rodičovského (abstraktního) případu užití, je uvedeno na obrázek 2.8. Rodičovský případ užití podle potřeby předává tok řízení specifickým případům užití (potomkům) v konkrétních bodech specifikace a z nich se tok řízení po provedení všech specifikovaných kroků činnosti předává zpět rodičovskému případu užití.



Obrázek 2.8 – Grafická vizualizace průběhu scénářů zobecněných případů užití v čase

Výklad

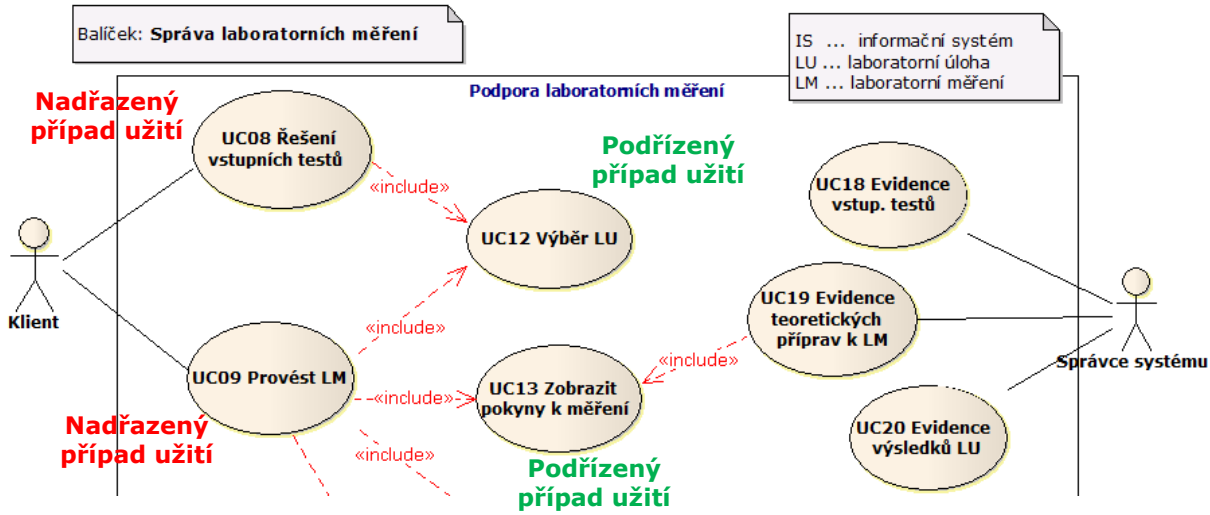
Rozšiřující relace závislostí případů užití

Pro zjednodušení diagramu případu užití, zavedení myšlenek objektových principů modelování a modelování funkcí prováděných za určitých podmínek a stavu systému se využívají další relace <<include>> a <<extend>> . Využití těchto relací musí však jednoznačně definovat systém, zjednodušit a zefektivnit celkový model sytému. Patří tedy do pokročilého modelování případu užití.

Relace <<include>>

Společné funkce modelovaného systému je vhodné definovat do společného případu užití s relací <<include>> navazující na ostatní případy užití, které využívají tento společný případ užití jako část své celkové činnosti (obrázek 2.9). Právě relace <<include>> označuje podřízené funkce nebo vlastnosti navazující na nadřazený (volaný) případ užití.

Modelování s vazbou <<include>> má výhody v objektovém přístupu k modelování informačního systému a jeho následné implementaci. Analytik tímto lépe definuje systém pro následnou implementaci objektů, metod a vlastností v objektovém programovém prostředí.



Obrázek 2.9 – Rozšiřující relace <<include>> závislostí případů užití

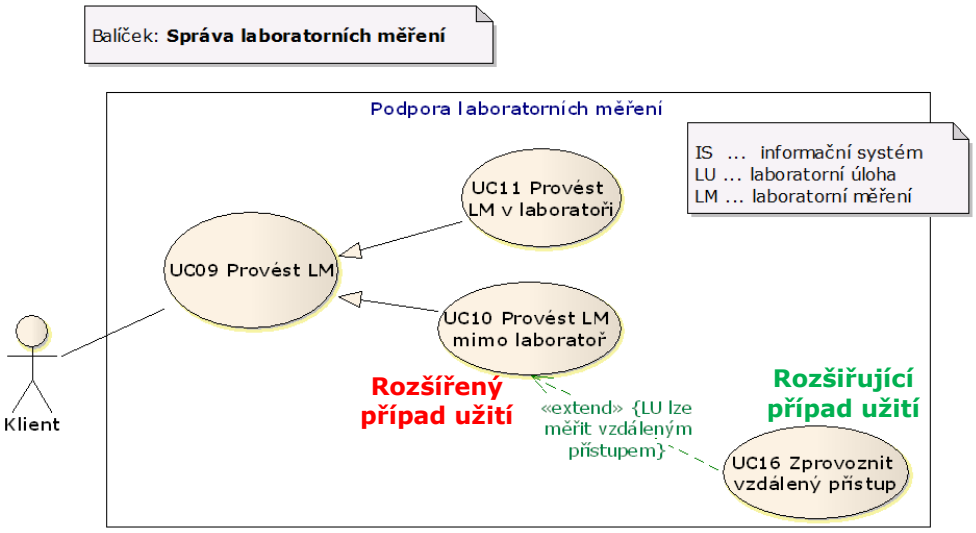
Ve scénářích nadřazených případů užití se objeví místo, kdy se zařazuje provedení scénáře podřízeného případu užití. Například v tab. 2.2 je v základním scénáři aplikována relace <<include>> hned v prvním bod, konkrétně „1. Zahrnout **UC12 Výběr LU**“. Po provedení celého scénáře podřízeného případu užití se pokračuje v dalším kroku scénáře nadřazeného případu užití.

Relace <<extend>>

Konkrétní činnost systému má definován případ užití, který může být po určité době rozšířen o další dodatečné funkce. Ať už z důvodu pozdějšího rozšíření modelu případu užití, nebo z důvodu použití této části funkcí i pro jiný případ užití. Rozšiřující případ užití vázaný relací <<extend>> navazuje v bodě rozšíření původního případu užití a je nutné při definování

scénářů definovat bod rozšíření uvedením jeho znění do složených závorek, např. {LU lze měřit vzdáleným přístupem}, a pro nový případ užití lze definovat vlastní vstupní i následné podmínky, které má rozšiřující případ užití splnit (obrázek 2.10).

Případ užití může být rozšířen jedním nebo i více případy užití, pokud pro rozšířený případ užití je analyzováno více podmínek, za nichž mají být provedeny rozšiřující případy užití. To už je tak specializované modelování diagramů případů užití, že zde ukázky již uvedeny nejsou a lze je najít např. v publikaci [Arlow, 2003].



Obrázek 2.10 – Rozšiřující relace <<extend>> závislostí případů užití

Ve scénářích rozšířených případů užití se vkládá relace <<extend>> stejně jako se definuje podmíněný krok scénáře. Nejlépe to vystihují scénáře obou zúčastěných případů užití rozšiřující relace <<extend>> uvedené v *tab. 2.5* a *tab. 2.6*.

Tab. 2.5 – Scénář případu užití rozšířeného případu užití UC10.

Název případu užití	**Provést LM mimo laboratoř (specifický UC)**
Identifikátor	UC10
Cíl	Nové poznatky, informace a data z LM.
Primární účastník	Klient
Sekundární účastník	Správce systému, Systém
Vstupní podmínky	Pro vybranou laboratorní úlohu *Klient* splnil vstupní test.
Výstupní podmínky	Dostupné soubory dat z LM.
Základní scénář	**v bodě {Specifikace provedení LM}** 1. Systém upozorní *Klienta*, že část aplikací musí uzavřít standardním způsobem uzavření aplikací Windows a vrátit se zpět k původnímu IS.

	2. Systém spustí aplikaci prohlížeče s parametrem URL adresy pro vybranou LU. 3. KDYŽ je možné LU měřit pouze s využitím přístup ke vzdálené ploše, pak **UC16 Zprovoznit vzdálený přístup** 4. Jinak: 4.1. Systém spustí IE s parametrem URL adresy podle vybrané LU. 4.2. Zahrnout **UC14 Naměřit data**. 4.3. Zahrnout **UC15 Zálohovat data**. 4.4. *Klient* ukončí IE standardním uzavřením Windows aplikace. 5. *Klient* předá informaci systému o ukončení činnosti v jiných aplikacích.
Alternativní scénář(e)	

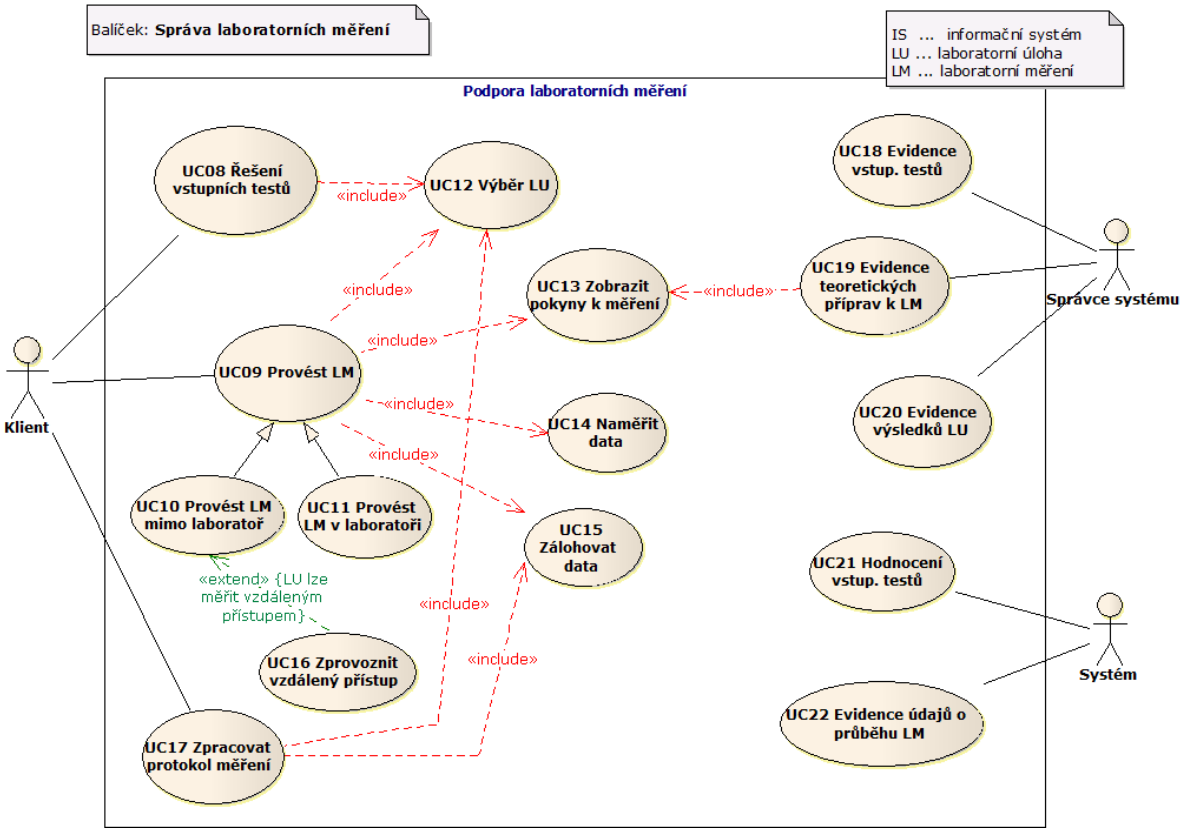
Tab. 2.6 – Scénář případu užití rozšiřujícího případu užití UC16.

Název případu užití	**Zprovoznit vzdálený přístup (rozšiřující UC)**
Identifikátor	UC16
Cíl	Provést zprovoznění vzdáleného přístupu pro realizaci úlohy LM.
Primární účastník	Klient
Sekundární účastník	Systém
Vstupní podmínky	LM má být provedeno mimo laboratoř.
Výstupní podmínky	Zajištění komunikace přes vzdálenou plochu.
Základní scénář	1. Systém zobrazí informace o dílčích krocích k zpřístupnění LU (IP adresa, název účtu a heslo na vyžádání Správce systému, název aplikace na ploše vzdáleného počítače). 2. Systém spustí aplikaci VPN Client. 3. Systém spustí aplikaci Remote Desctop. 4. *Klient* se přihlásí v aplikaci VPN Client. 5. *Klient* se přihlásí na účet vzdáleného počítače. 6. Zahrnout **UC14 Naměřit data**. 7. Zahrnout **UC15 Zálohovat data**. 8. *Klient* ukončí aplikaci standardním způsobem Windows. 9. *Klient* ukončí přístup k vzdálené ploše počítače.
Alternativní scénář(e)	

Řešený příklad

V balíčku Správa laboratorních měření existuje případ užití **Provést LM** (laboratorní měření), který se dále člení na další případy užití: **Výběr LU**, **Zobrazit pokyny k měření**, **Naměřit data** a **Zálohovat data**. Mezi nimi existuje vazba <<include>>, která se zavádí tehdy, pokud jsou některé činnosti užívány i jinými případy užití. Pak se nemusí zavádět tyto případy užití vícekrát. Pokud by neexistovala návaznost z případu užití **Evidence teoretických příprav k LM** nebo z případu užití **Zpracovat protokol měření**, pak by určitě bylo nevhodné pro modelovaný systém zavádět vazbou <<include>> vložené případy užití. Stačí pak činnosti těchto podřízených případů užití zahrnout do scénáře nadřazeného případu užití **Provést LM**.

Využita je v ukázkovém příkladu také vazba <<extend>> mezi případy užití **Provést LM mimo laboratoř** a **Zprovoznit vzdálený přístup**. Rozšiřující případ užití **Zprovoznit vzdálený přístup** je proveden jedině za splnění dané podmínky, která je uvedena jako specifikace vazby <<extend>>, tj. {LU lze měřit vzdáleným přístupem}. Pokud *Klient* nepotřebuje zprovoznění vzdáleného přístupu pro měření vybrané úlohy, pak tento případ užití nebude proveden. Na základní úrovni tvorby případů užití je rovněž možné zahrnout scénář rozšiřujícího případu užití do scénáře rozšířeného případu užití, ale tím se jeho velikost značně rozšíří a znepřehlední.



Obrázek 2.11 – Ukázka aplikací rozšiřujících relací závislostí případů užití

Pokud bychom to porovnali např. s tvorbou programového kódu, tak je z hlediska objektových principů programování distribuovat částí zdrojových kódů do jednotlivých funkcí, nikoliv tvořit jednu funkci s příliš dlouhým zdrojovým kódem.

Shrnutí pojmů

Správa požadavků je základním kamenem neboli první částí návrhu modelovaného systému. Je důležité stanovit hranice modelovaného systému a jeho konkrétní požadavky, které budou dále implementovány. Vždy je důležité vědět, co od systému očekávám, jak ze strany zadavatele, tak ze strany řešitele modelovaného systému.

Analytik navrhuje požadavky systému a konzultuje tyto požadavky vždy se zadavatelem a také všemi účastníky, kteří se podílí na vzniku daného systému. Až po dosažení shody mezi zadavatelem a analytikem může být navržený model v rámci požadavků dekomponován a převeden do implementačního prostředí a řádně testován uživateli systému.

Kapitola požadavků modelovaného systému zahrnuje striktní popis diagramu případů užití včetně scénářů a jejich struktury dané notací UML. Studenti budou po přečtení kapitoly seznámeni s pojmy balíček, vazba, účastník, případ užití, scénář a další. Budou schopni nejen pochopit předložený diagram případů užití ve vztahu ke konkrétnímu modelovanému systému, ale také samostatně analyzovat požadavky modelovaného systému, navrhovat diagramy případů užití a efektivně sestavovat jejich scénáře.

Otázky

1. Jaké požadavky na modelovaný systém spadají do skupiny funkčních a jaké do skupiny nefunkčních požadavků?
2. Jaký název zvolíte pro účastníky modelovaného systému? Existují takové osoby nebo konkrétní systémy, nebo jsou to názvy fiktivní?
3. Co zahrnují názvy a činnosti případů užití? Kdy už zvolit pro danou činnost samostatný případ užití a kdy danou činnost zahrnout do popisu scénáře jako dílčí bod?
4. Jaký je přínos pojmu „zobecnění“ v oblasti analýzy a také implementace modelovaného systému?
5. Vysvětlete rozdíl v aplikaci vazby <<extend>> a <<include>>?
6. Co představuje pojem: „**v bodě {Specifikace provedení LM}**“ v hlavní části scénáře?
7. Kdy využíváme alternativní scénáře pro konkrétní případ užití? Kolik alternativních scénářů může mít případ užití obecně (*možnosti*: 0 nebo 1 nebo 2 nebo neomezený počet)?

3 REALIZACE POŽADAVKŮ MODELOVANÉHO SYSTÉMU

Čas ke studiu: 1 ½ hodiny

Cíl: Po prostudování tohoto odstavce budete umět

- Definovat způsoby realizace požadavků modelovaného systému.
- Rozhodnout o způsobu realizace v konkrétním případě.
- Navrhovat a realizovat sekvenční diagram.
- Navrhovat a realizovat komunikační diagram.
- Aplikovat a navrhovat diagramy aktivit v rámci modelovaného systému dle potřeby specifikace návaznosti dané činnosti.

Výklad

3.1 Realizace případů užití

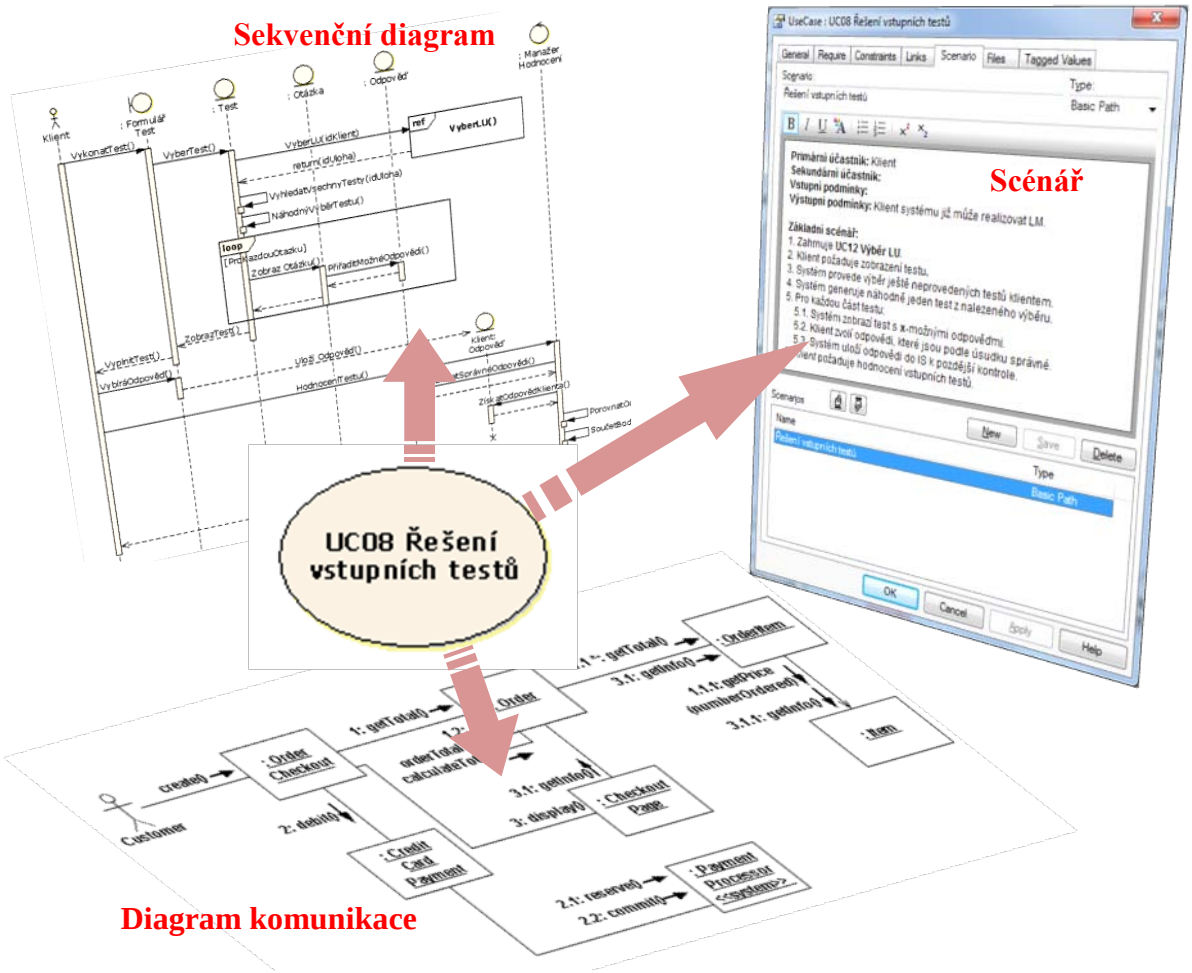
Specifikace činnosti případu užití se definuje scénářem k danému případu užití. V některých případech je nezbytné doplnit tyto scénáře podrobnějším popisem prostřednictvím sekvenčního diagramu, který má mnohem větší vizualizační schopnosti v časové posloupnosti jednotlivých činností, ale také nabízí nový pohled na případ užití v tom, že upřesňuje objekty (instance) mezi nimiž probíhá komunikace.

Analytici mají pro realizaci případů užití k dispozici tři nástroje modelovacího jazyka UML (obrázek 3.1), které mohou použít pro specifikaci případů užití, konkrétně pro současnou verzi UML2.3 se tyto nástroje nazývají

- scénáře případů užití,
- sekvenční diagramy,
- diagramy komunikace.

Scénáře byly popsány již v předchozích kapitolách společně s popisem modelování případů užití (kapitola 2.4) i v následné fázi, tj. pokročilé modelování případů užití (kapitola 2.5). Scénář slovně definuje činnost případu užití a je *primárním nástrojem* pro modelování případů užití a jejich realizaci. Existence scénářů je tedy zásadním předpokladem v modelování případů užití a jejich následující realizace.

Ze scénářů není zřejmá časová závislost a bližší interakce mezi objekty (instancemi, třídami). K tomu jsou určeny sekvenční diagramy nebo diagramy komunikace (neobsahují pojetí časové závislosti). Převážně se v modelování interakcí využívá spíše sekvenčních diagramů. Následující kapitoly se budou zabývat popisem sekvenčních diagramů a diagramů komunikace.



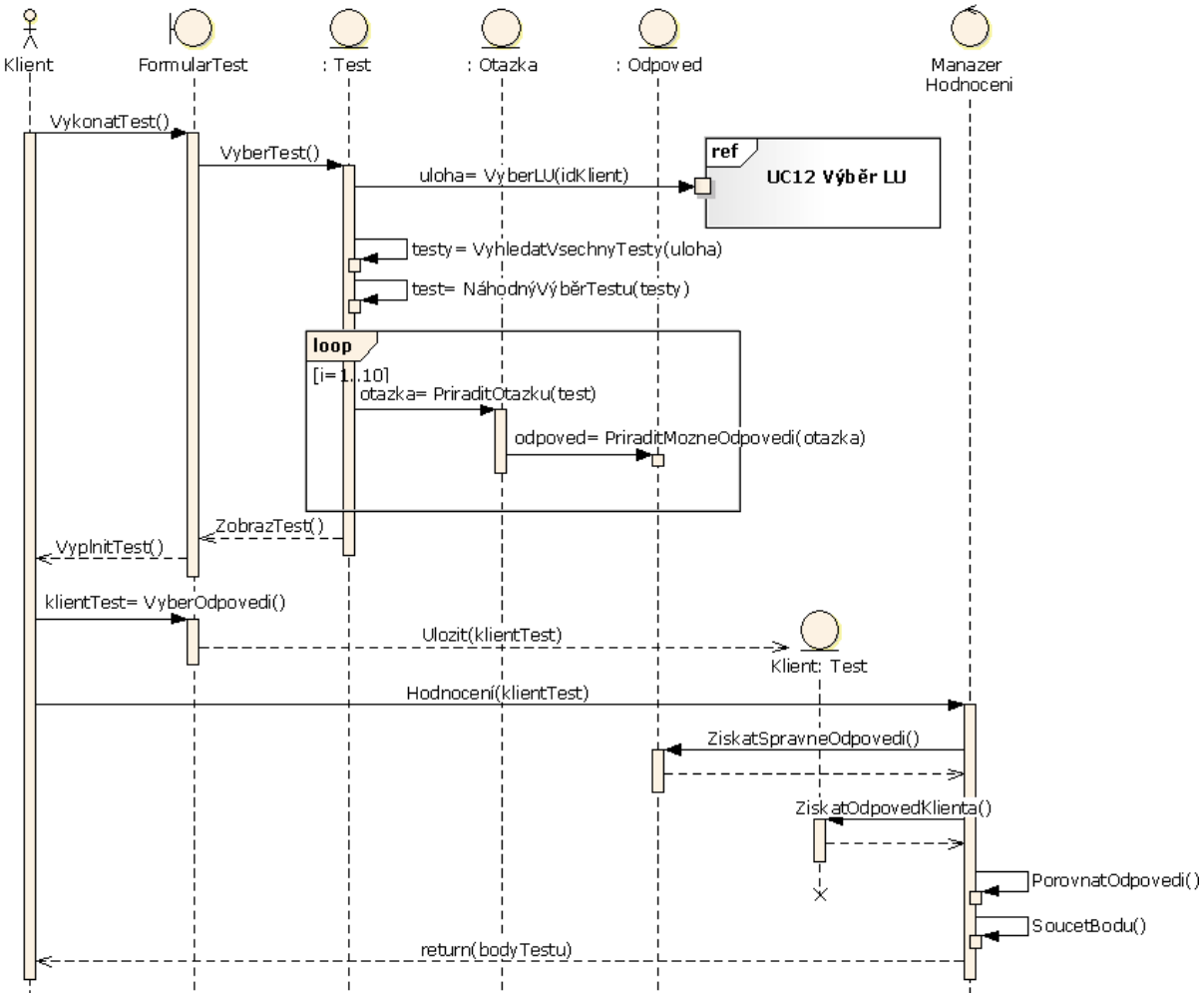
Obrázek 3.1 – Způsoby realizace případu užití

Výklad

3.2 Sekvenční diagramy

Sekvenční diagramy definují uspořádané vyjádření interakcí mezi jednotlivými objekty, tak i třídami či dokonce účastníky jako sérii následných kroků v průběhu času (obrázek 3.1). Jsou využívány zejména tam, kde v modelovaném systému je kromě vazeb komunikačních také důležitá časová souslednost modelovaného procesu. Prvky, které zde mezi sebou komunikují (účastníci, třídy, objekty) se souhrnně nazývají klasifikátory.

V zjednodušeném pojetí sekvenční diagramy představují předávání zpráv od jednoho klasifikátoru k druhému v horizontální linii. Zpráva, která je umístěny výše než jiná zpráva, v čase proběhne dříve. Z každého klasifikátoru (lépe řečeno jeho instance) směřuje dolů ve vertikálních liniích tzv. čára života, na níž se graficky zobrazuje. Zahájení toku událostí mezi klasifikátory navzájem je aktivováno účastníkem.



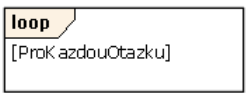
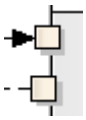
Obrázek 3.2 – Sekvenční diagram případu užití UC08 Řešení vstupních testů

Rozhodně se návrh sekvenčních diagramů provádí z již dříve zpracovaného scénáře případu užití, jehož činnost je z důvodu zřetelnějšího pochopení dané činnosti vhodnější specifikovat také sekvenčním diagramem. Není vhodné navrhovat sekvenční diagramy pro jednoduché činnosti, které stačí specifikovat jen scénářem. Jak již bylo výše uvedeno, cílem modelování není provádět zbytečně složité analýzy informačních systémů, ale jednoduché a obsahem postačující souhrny diagramů přesně a striktně vystihující modelovanou problematiku. Pro zobrazení sekvenčních diagramů se využívají prvky uvedené v tab. 3.1, kde je uveden jejich grafický symbol a také popis.

Základní scénář:
1. Zahrnuje **UC12 Výběr LU**.
2. Klient požaduje zobrazení testu.
3. Systém provede výběr ještě neprovedených testů klientem.
4. Systém generuje náhodně jeden test z nalezeného výběru.
5. Pro každou část testu:
   5.1. Systém zobrazí test s **x**-možnými odpověďmi.
   5.2. Klient zvolí odpovědi, které jsou podle úsudku správné.
   5.3. Systém uloží odpovědi do IS k pozdější kontrole.
6. *Klient* požaduje hodnocení vstupních testů.

Obrázek 3.3 – Základní scénář případu užití UC08 Řešení vstupních testů

Tab. 3.1 – Grafická syntaxe a popis prvků sekvenčního diagramu

Prvky sekvenčního diagramu	Popis
Účastník (Actor) 	Účastníci reprezentují roli uživatele v sekvenčním diagramu. Dalším typem objektu, který je určen ke komunikaci, je třída nebo její instance označené obecným prvkem objekt. Tento prvek je součástí životní čáry. Pokud je požadavek zavést nový objekt v sekvenčním diagramu, pak se tato činnost realizuje tvorbou prvku „životní čára“.
Životní čára (Lifeline) 	Životní čára je grafickým vyjádřením účasti daného objektu v sekvenčním diagramu. Životní čára je značena svislou čárkovanou čarou směřující dolů od daného účastníka nebo objektu.
Aktivita objektu (Focus of Control) 	Prvek vyznačuje aktivní účast v daném čase v sekvenčním diagramu. Toky řízení vždy začínají nebo končí na některé životní čáře, respektive aktivitě objektu, v sekvenčním diagramu. Vyznačuje se obdélníkem, který překrývá životní čáru. Účastník je aktivní tehdy, pokud od prvku aktivita objektu směřuje řídicí tok k jinému účastníku nebo objektu, respektive jeho aktivitě objektu.
Hraniční třída (Boundary) 	Třídy, které se využívají ke komunikaci systému a uživatelů na úrovni specifického typu rozhraní, např. obrazovka, formulář, komunikační protokol, rozhraní pro tiskárnu. Označují se stereotypem <<boundary>>.
Třída řídicí (Control) 	Prvek Řízení je třída, která koordinuje chování v systému prostřednictvím provedení různých činností a operací. Třída Control obvykle má chování, které je specifické pro jeden případ užití a obvykle objekt zaniká s ukončením realizovaného případu užití. Používají se k nastavení obsahu entitních tříd.
Entitní třída (Entity) 	Jedná se o pasivní třída neboli objekt, který slouží jako paměť nebo úložný mechanismus, který zachycuje znalosti nebo informace v systému na delší dobu. Často odpovídají objektům reálného světa (klient, účet apod.)
Rámec (Fragment) 	Interakční rámec, které umí definovat činnost typu iterace, selekce apod. V levém horním rohu je vyznačeno klíčové slovo (loop, alt, opt, break, seq, strict apod.) podle typu rámce pro implementaci iterace, selekce, rámec s podmínkou, atd.
Uvolnění objektu (Endpoint) 	Ukončuje existenci daného objektu. Je umístěn na životní čáře tohoto objektu a v dané horizontální linii se předpokládá, že jeho využití v sekvenčním diagramu končí. Běžně jsou životní čáry vedeny dolů níže než je úroveň poslední zprávy v řídicím toku a končí u všech účastníků nebo objektů na stejné horizontální úrovni.
Brána (Diagram Gate) 	Brána je vizuální indikací bodu, kde zpráva prostupuje do/z konkrétního rámce, který je součástí sekvenčního diagramu. Graficky je brána značena prázdným čtvercem umístěným na okraj rámce.

Prvek *Rámec* je určen pro několik různých činností, které specifikují daný tok řízení v sekvenčním diagramu. Jeho specifikace je dána především volbou interakčního operátoru, který je zobrazen zkratkou v levém horním rohu prvku *Rámec*. Seznam často využívaných interakčních operátorů a jejich stručný popis je uveden v tab. 3.1. Další operátory: *critical*, *neg*, *assert*, *strict*, *seq*, *ignore* a *consider* nebudou v této publikaci popsány, neboť jsou příliš specifické pro tento studijní podklad.

Tab. 3.2 – Seznam možných parametrů rámce (Fragment) sekvenčního diagramu

alt	**Alternativně podmíněné vykonávání zpráv**. Rámec s označením *alt* se skládá z více podmíněných částí, z nichž každá část obsahuje posloupnost zpráv provedených pouze při splnění podmínky individuální pro každou část rámce. Podmínka je uvedena v hranatých závorkách.
opt	**Podmíněné vykonávání zpráv**. Posloupnost zpráv uvedených v rámci s interakčním parametrem *opt* se vykoná pouze tehdy, je-li splněna podmínka uvedená uvnitř hranatých závorek.
par	**Paralelní zpracování vykonávaných zpráv**. Rámec může mít dvě i více paralelních částí. Následující zprávy vykonávané za tímto rámcem budou uskutečněny po provedení zpráv ve všech paralelních větvích.
loop	**Opakování vykonávaných zpráv**. Rámec je vykováván podle podmínky uvedené v hranatých závorkách. Opakování může být dáno pevným počtem (ve scénáři klauzule PRO) nebo podmínkou (DOKUD).
break	**Podmíněné vykonávání zprávy s přerušením**. Podobná varianta jako rámec s parametrem *opt*, kromě jiného názvu a toho, že po splnění podmínky se po provedení všech zpráv uvedených v rámci přeruší činnost vykonávání dalších zpráv.

Objekty sekvenčního diagramu již byly popsány, ale chybí popis syntaxe komunikačních zpráv mezi nimi. Zpráva aktivující činnost označována plnou čarou zakončenou šipkou ve směru toku zprávy. Provádění konkrétní činnosti reprezentuje prvek Aktivita objektu, který znázorňuje dobu trvání činnosti. Návratové zprávy se zobrazují čárkovanou čarou a doporučují se vkládat pouze tehdy, když je zapotřebí zdůraznit parametr, který vrací. Jinak pro lepší přehlednost se vkládat návratové zprávy nemusí.

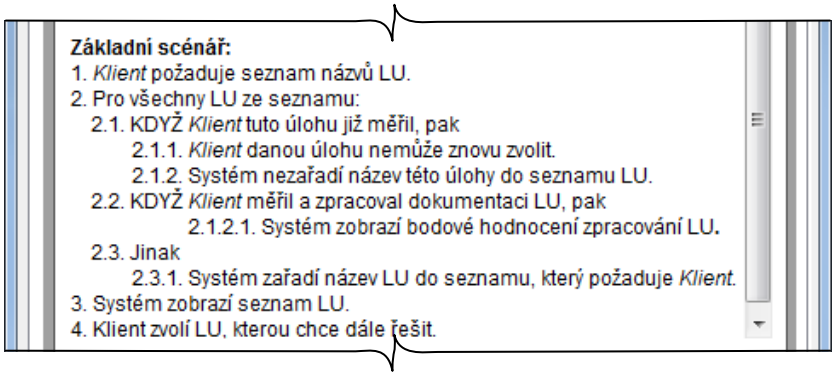
Zprávy, které se využívají k vytvoření objektu (třídy, instance) jsou rovněž zobrazeny čárkovanou čarou, někdy se též zpráva označuje klíčovým slovem <<create>>. Zprávy, které jsou určeny pro odstranění objektu, se zobrazují plnou čarou a doporučuje se vkládat klíčové slovo <<destroy>>.

Řešený příklad

Tvorba sekvenčního diagramu (obrázek 3.2) byla založena na činnosti případu užití **UC08 Řešení vstupních testů**. Nejprve byl sestaven scénář k danému případu užití a tvorba sekvenčního diagramu je specifikací a zpřesněním vykonávání zpráv daných konkrétním případem užití.

Při návrhu sekvenčního diagramu se v zásadě postupuje z levého horního rohu směrem dolů nebo vpravo. Základní scénář je dán sekvencemi činností uvedenými ve scénáři na obrázek 3.3. Scénář má pouze dva účastníky *Klient* a *Systém*, přičemž pod pojmem *Systém* není čistě chápán účastník, ale modelovaný informační systém. Při návrhu sekvenčního diagramu a jeho nezbytných klasifikátorů se pojem *Systém* modeluje jedním z klasifikátorů: hraniční třída, entitní třída a třída řídicí (obrázek 3.2).

Scénáře vždy vychází z aktivity vyvolané účastníkem. Existují pravidla při sestavení sekvenčního diagramu taková, že **entitní třída není přímo nikdy vázána na účastníka**. Entitní třída může představovat datové údaje, které jsou uloženy v informačním systému. Aby požadované datové údaje bylo možné z informačního systému získat a předat účastníku, pak musí být v komunikační zprávě mezi účastníkem zahrnuta třída řídicí (pro získání dat z IS) a hraniční třída (komunikační rozhraní mezi účastníkem a systémem).



Obrázek 3.4 – Základní scénář případu užití UC12 Výběr LU

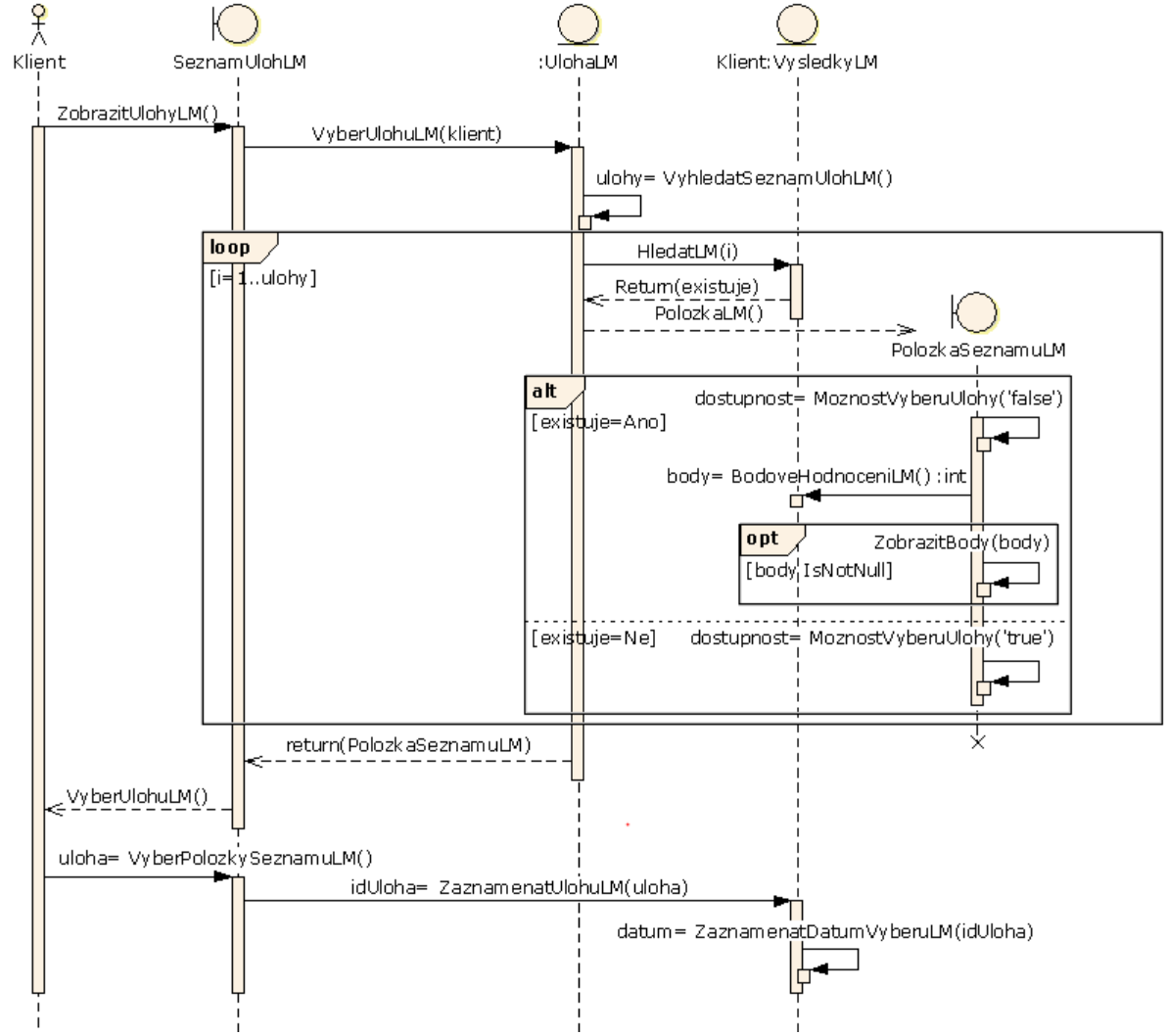
První činností scénáře je provést scénář jiného případu užití (**UC12 Výběr LU**), viz jeho scénář (obrázek 3.4) a sekvenční diagram (obrázek 3.5). První dva body ze scénáře představují v sekvenčním diagramu první tři zprávy (**VykonatTest**, **VyberTest** a **VyberLU**) a do vykonávání je zahrnut rámec s operátorem *ref* s grafickými prvky brány na hranici rámce. Třetí i čtvrtý krok scénáře je modelován zprávou, která se vyznačuje tím, že objekt, který ji volá, ji zároveň také provádí. U zprávy **VyhledatVsechnyTesty**(uloha) by měla být podmínka, která zajistí, že test, který již *Klient* provedl neúspěšně, nemůže být znovu vybrán. Pátý krok scénáře představuje prvek Rámec, v němž se provádí příslušné činnosti. Poslední krok je ve scénáři zapsán obecně a sekvenční diagram jej vystihuje vykonáním více než jedné zprávy, což je důsledkem toho, že entitní třída není vázána přímo na účastníka, ale prostřednictvím hraniční třídy, případně také třídy řídicí.

V sekvenčním diagramu je možné vidět tři typy rámců, které se odlišují konkrétním označením **loop**, **alt** a **opt**. Uvnitř rámců jsou definovány podmínky rámce podle jeho vykonávané činnosti.

Rámec **loop** definuje cyklus prostřednictvím podmínky [*i* = 1..ulohy], kde *i* je název cyklicky opakujícího se parametru, který se vyskytuje v definici názvu zprávy uvnitř rámce, např. *HledatLM*(*i*).

Rámec **alt** je určen k vyhodnocení podmínky, proto je vertikálně rozdělen do dvou částí, kdy zprávy jsou vykonávány dle té části podmínky, která je splněna. Pak je seznam vybraných laboratorních úloh (*PolozkaSeznamuLM*) dán těmi úlohami, které **nebyly** měřeny a zároveň bodově zhodnoceny.

Poslední rámec uvedený v obrázku (obrázek 3.5) je rámec **opt**, kde je uvedena podmínka [body IsNotNull]. Zpráva **ZobrazitBody(body)** uvnitř rámce **opt** jsou vykonány jedině při splnění podmínky, že hodnota parametru body není nulová.



Obrázek 3.5 – Sekvenční diagram případu užití UC12 Výběr LU

Značení podmínek není dle notace UML syntakticky definováno, nicméně značení by mělo odpovídat notaci implementačního prostředí, pro které je analyticky navržen tento sekvenční diagram.

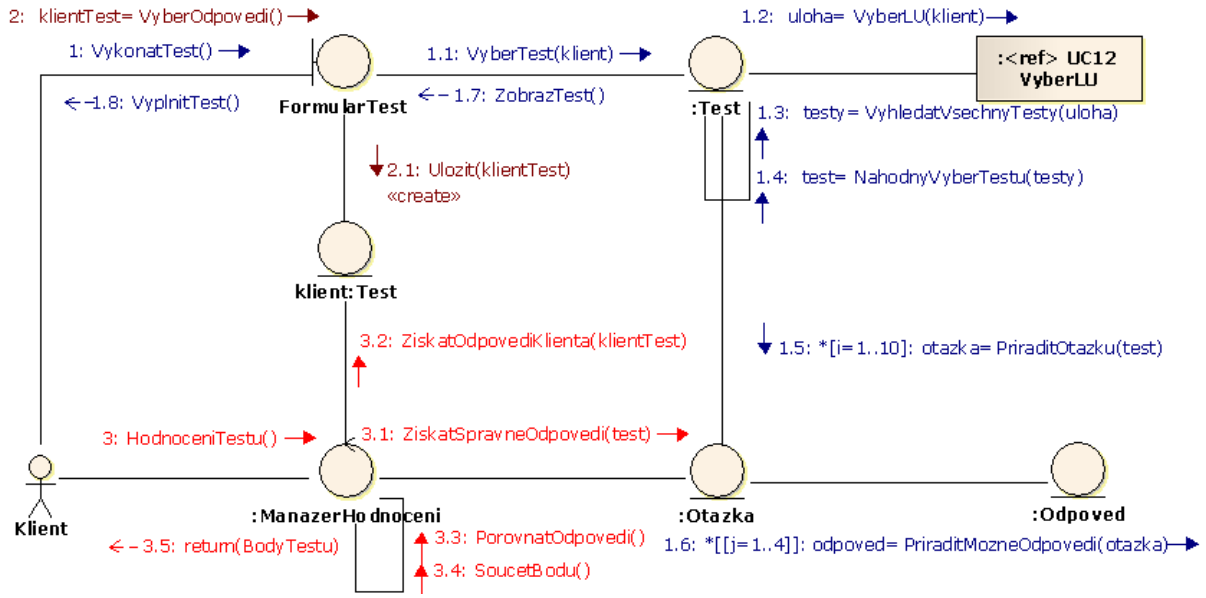
Ačkoliv ve scénáři je uveden pouze účastník *Klient* a pojem Systém, který zde nezastupuje účastníka aktivujícího případ užití, ale pouze modelovaný systém. V sekvenčním diagramu pak posléze přibývají konkrétní objekty (*instance, třídy*) s konkrétními názvy a to

už je analytik zase o krok dál při tvorbě analýzy systému, který směřuje dál na objektové modelování.

Výklad

3.3 Diagramy komunikace

Diagram komunikace společně se sekvenčním diagramem patří do skupiny procesních diagramů, které jsou určeny pro vizualizaci interakcí mezi jednotlivými objekty. Zatímco sekvenční diagramy jsou aplikovány pro specifikaci interakcí mezi objekty v čase, diagramy komunikace se zaměřují pouze na zobrazení interakcí mezi objekty a jsou organizovány v závislosti na prostoru. Časový rozměr v diagramu komunikace je znázorněn pouze číslováním posloupnosti zpráv mezi objekty (obrázek 3.6). Víceúrovňové číslování zdůrazňuje činnosti aktivované v daném čase zpravidla účastníkem. Návaznost zpráv vyjadřuje posloupnost číslování v druhé a nižších úrovních. Návratové zprávy se značí čárkovanou čarou se šipkou. Analýza diagramů komunikace dle standardu UML využívá prvky uvedené v tab. 3.3, kde je uveden jejich grafický symbol a také popis.



Obrázek 3.6 – Diagram komunikace případu užití UC08 Řešení vstupních testů

Specifikace řízení toku činností v diagramu komunikací

Řízení toku činností (zpráv) v diagramu komunikací je způsobeno omezením podmínky pro vykonání zprávy nebo cyklickým vykonáváním zpráv. Syntaxe pro **podmíněné vykonání činností**, **zpracování činností v cyklu s pevným nebo proměnným počtem opakování** zpráv je vzhledem k sekvenčním grafům jednotnější.

Syntaxe **podmíněných činností** v celé zprávě je uvedena takto:

pořadČíslo: [*podmínka*] *navratHodnota=nazevZpravy*(*parametr*).

Následující zprávy budou vykonány pouze tehdy, když bude splněna podmínka v nadřazené zprávě (např. pro zprávu 1.3 [*pocet*>5] NajitProdukt(), jsou vykonány zprávy 1.3.1. ZjistitNazev(), 1.3.2. ZjistitCenu(), atd.).

Cyklus s pevným nebo proměnným počtem opakování a jeho podřízené zprávy, které jsou označeny nižší úrovní číslování, je uveden syntaxí:

pořadČíslo: *[*cyklus*] *navratHodnota=nazevZpravy*(*parametr*)

Konkrétní výrazy pro zastupující označení *cyklus* jsou dány např. podmínkou [*n*=1..10] pro cyklus s pevným počtem opakování od 1 do 10. Pro cyklus s proměnným počtem opakování neboli cyklem zpracovávaným pouze za předpokladu splnění podmínky uvedené v hranatých závorkách, např. [*pocet*<10]. Takový cyklus bude prováděn tak dlouho, dokud hodnota proměnné *pocet* bude menší než 10.

Dalším základním požadavkem na řízení toku činností je **větvení zpracování zpráv**, které je dáno syntaxí:

pořadČíslo: [*podmínka*] *nazevZpravy(parametr)*

Zde se navíc k úrovním číslování přidává další značení písmeny abecedy. Takže pro první větev, kdy bude splněna podmínka pro činnost

1. [*platba*='karta'] ProvestUhradu()

budou po splnění této podmínky vykonávány zprávy s označením 1a.1., 1a.2., atd. Pokud není splněna podmínka, budou prováděny zprávy s označením 1b.1., 1b.2., a další.

Tab. 3.3 – Grafická syntaxe a popis prvků diagramu komunikace

Prvky diagramu komunikace	Popis
Účastník (Actor) 	Účastníci reprezentují roli uživatele v sekvenčním diagramu. Dalším typem objektu, který je určen ke komunikaci, je třída nebo její instance označené obecným prvkem objekt. Tento prvek je součástí životní čáry. Pokud je požadavek zavést nový objekt v sekvenčním diagramu, pak se tato činnost realizuje tvorbou prvku „životní čára“.
Objekt (Object) 	Objekt je konkrétní instancí třídy. Objekty se často využívají v analýze, aby zde zastupovaly četné artefakty nebo předměty, např. papíry, faxy a informace. Prvek Objekt nabízí řadu stereotypů, mimo jiné i stereotypy označující následující tři prvky této tabulky. Při změně stereotypu se změní také tvar prvku Objekt.
Hraniční třída (Boundary)	Třídy, které se využívají ke komunikaci systému a uživatelů na úrovni specifického typu rozhraní, např. obrazovka, formulář, komunikační protokol, rozhraní pro tiskárnu. Označují se stereotypem <<boundary>>.

	
Třída řídicí (Control) 	Prvek Řízení je třída, která koordinuje chování v systému prostřednictvím provedení různých činností a operací. Třída Control obvykle má chování, které je specifické pro jeden případ užití a obvykle objekt zaniká s ukončením realizovaného případu užití. Používají se k nastavení obsahu entitních tříd.
Entitní třída (Entity) 	Jedná se o pasivní třídu neboli objekt, který slouží jako paměť nebo úložný mechanismus, který zachycuje znalosti nebo informace v systému na delší dobu. Často odpovídají objektům reálného světa (klient, účet apod.)

V některých diagramech komunikace se místo tříd řídicích nebo hraničních či entitních tříd využívají prvky Objekt. Prvky stejného vzhledu na první pohled nevyjadřují konkrétní typ prvku a pochopení takového diagramu je náročnější. Proto je vhodnější využívat těchto různorodých vizualizačních prvků pro jednoznačnost vyjádření typu konkrétní entity.

Objekty diagramu komunikací se běžně propojují standardními asociačními vazbami a směr zpráv, kterých může být mezi dvěma objekty více, jsou odlišeny nejenom číslováním, ale také směrem šipky zobrazené u každé zprávy.

Řešený příklad

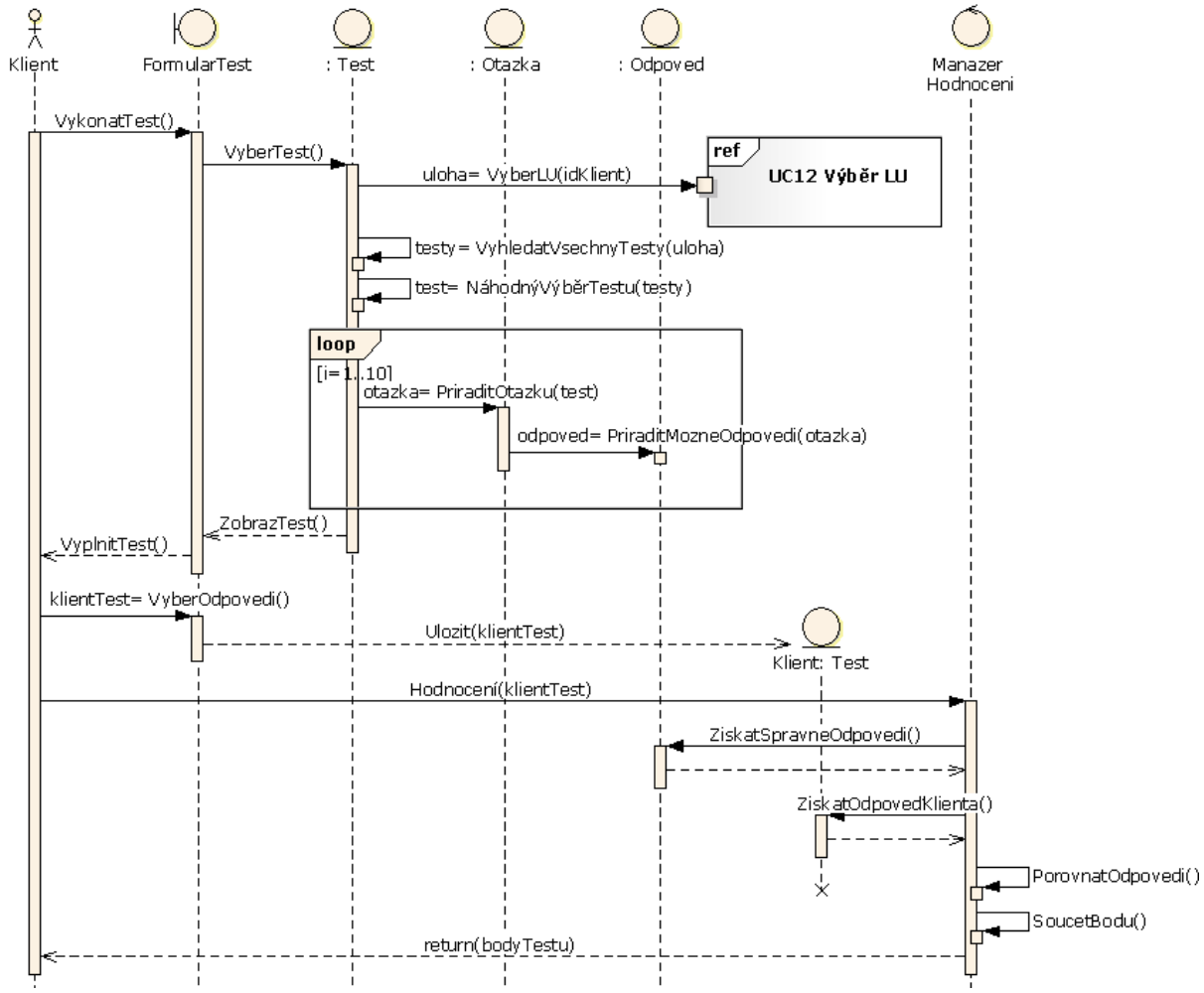
Diagram komunikace je jinou variantou interaktivního diagramu, proto pro porovnání obou typů diagramů, tj. *sekvenčního diagramu* (obrázek 3.5) a *diagramu komunikace* (obrázek 3.6), byly uvedeny ukázky diagramů pro stejný případ užití **UC08 Řešení vstupních testů**. Na těchto diagramech je možné si porovnat aplikaci prvků odkazu na jiný případ užití, vyjádření časové posloupnosti zpráv a označení cyklických a podmínečných zpracování činností.

Vycházíme v tomto příkladu ze scénáře případu užití **UC08 Řešení vstupních testů** (obrázek 3.7). Při návrhu komunikačních diagramů se zpravidla vychází ze scénáře případu užití. V tomto učebním textu je v analýze využíván standardně sekvenční diagram, proto i v tomto řešeném příkladu bude cílem ukázat rozdíl mezi komunikačním a sekvenčním diagramem.

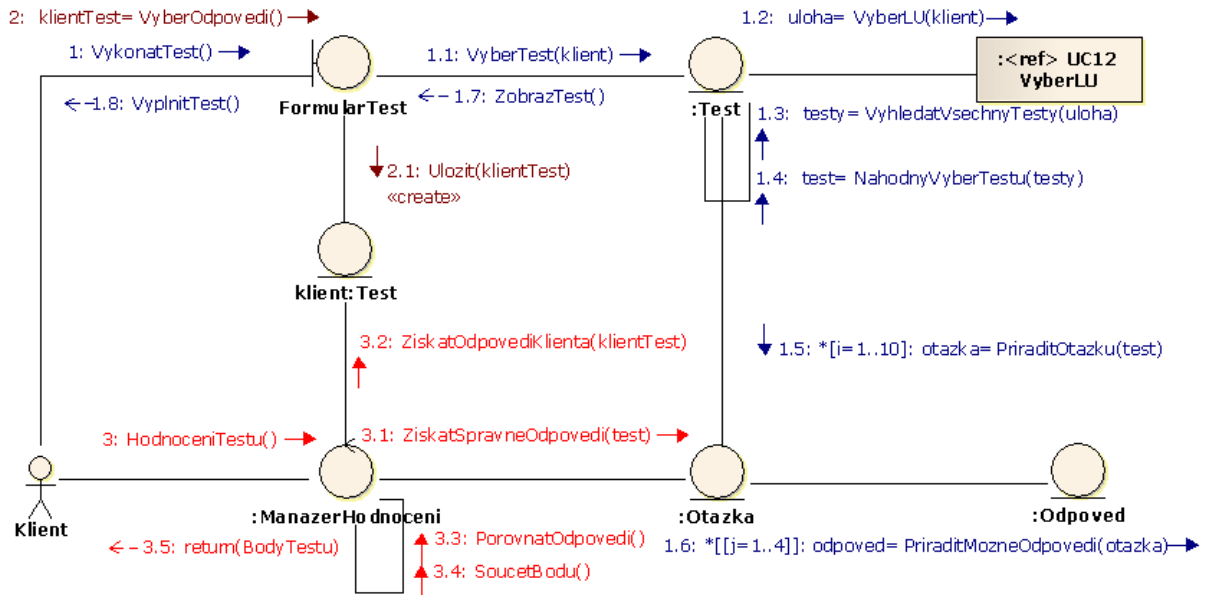
Základní scénář:
1. Zahrnuje **UC12 Výběr LU**.
2. Klient požaduje zobrazení testu.
3. Systém provede výběr ještě neprovedených testů klientem.
4. Systém generuje náhodně jeden test z nalezeného výběru.
5. Pro každou část testu:
   5.1. Systém zobrazí test s **x**-možnými odpověďmi.
   5.2. Klient zvolí odpovědi, které jsou podle úsudku správné.
   5.3. Systém uloží odpovědi do IS k pozdější kontrole.
6. *Klient* požaduje hodnocení vstupních testů.

Obrázek 3.7 – Základní scénář případu užití UC08 Řešení vstupních testů

Vycházíme tedy z již navrženého sekvenčního diagramu, který již zde byl zobrazen, ale pro porovnání jej tedy znovu zobrazíme i zde v této kapitole (obrázek 3.8). Sekvenční diagram zobrazuje tři základní zprávy aktivované účastníkem *Klient*: **VykonatTest**, **VyberOdpovedi**, **Hodnocení**. Každé z těchto aktivovaných zpráv odpovídá sekvence zpráv v diagramu komunikací odlišených barvou a v nejvyšší úrovni číslování.



Obrázek 3.8 – Sekvenční diagram případu užití UC08 Řešení vstupních testů



Obrázek 3.9 – Diagram komunikace případu užití UC08 Řešení vstupních testů

Modré zprávy v úrovni číslování 1 odpovídají sekvenci zpráv navazujících na aktivovanou zprávu **VykonatTest**. **Tmavě červenou barvou s číslováním v hlavní úrovni 2** jsou označeny zprávy, které navazují na činnost **VyberOdpovedi**. Poslední sekvencí zpráv zobrazených **výrazně červenou barvou s číslováním začínajícím číslem 3** jsou činnosti odpovídající aktivované zprávě **Hodnocení**.

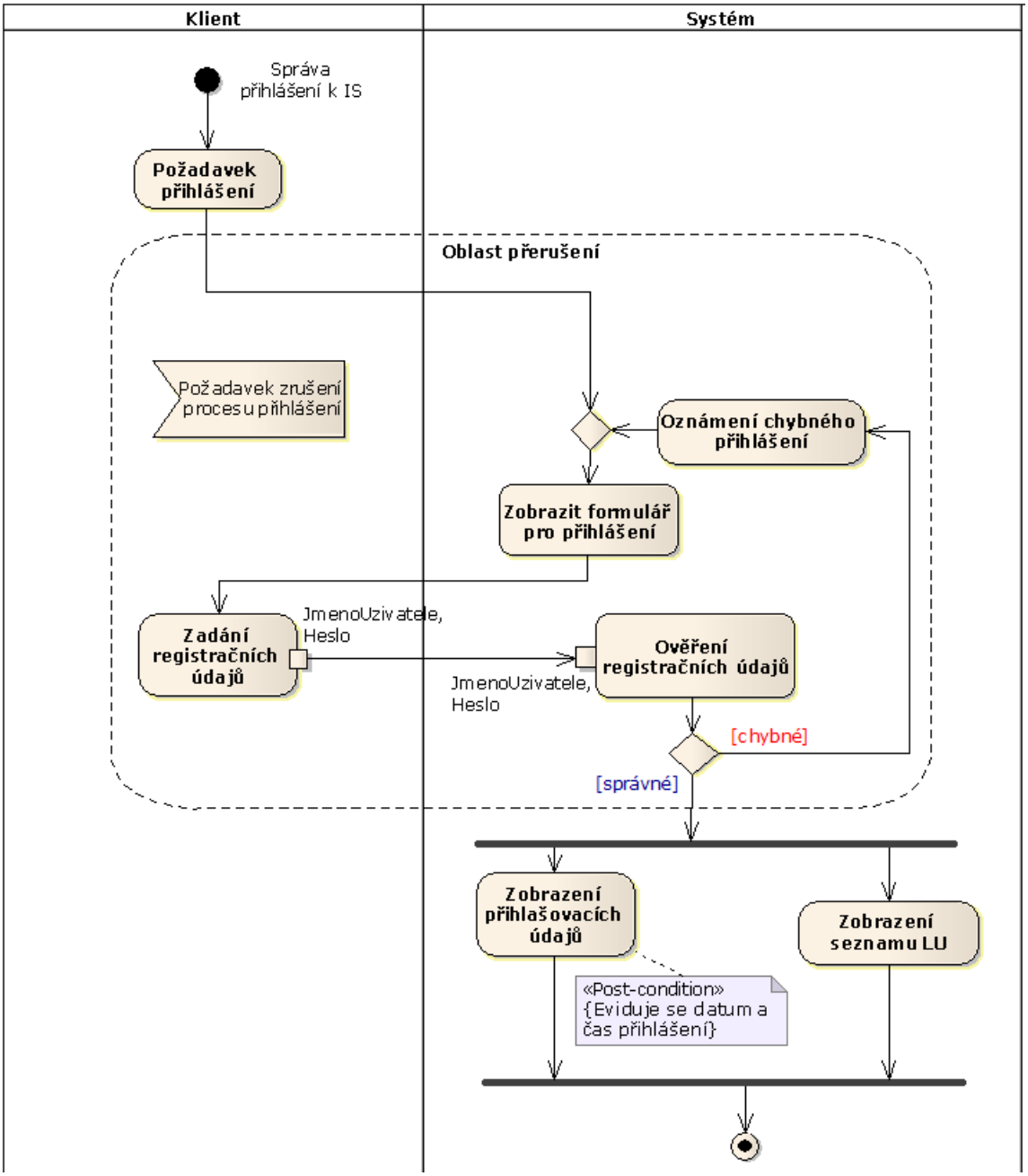
Stejně jako u sekvenčních diagramů, ani diagramy komunikační *nemusí* dle standardu UML *zobrazovat návratové zprávy*, které diagram mohou spíše znepřehlednit. Další výhodou sekvenčních diagramů je mimo jiné lepší viditelnost vzniku i zániku tříd, např. **Klient:Test**.

3.4 Diagramy aktivit

V UML se diagramy aktivit využívají k zobrazení procesů, jejich logiky a toku událostí detailně od počátečního ke koncovému uzlu. Diagramy aktivit jsou často přirovnávány k vývojovým diagramům, ale jejich možnosti jsou větší, protože navíc umožňují modelovat paralelní procesy.

Jejich činnost je rovněž řízena možným přerušením ze strany systému nebo jiné příchozí události. Činnosti, které mohou být při svém provádění přerušeny se zahrnují do ohraničeného celku s názvem Oblast přerušení. Pak je v oblasti přerušení definován prvek, jehož název vyjadřuje činnost, při které může dojít k přerušení všech činností v Oblasti přerušení.

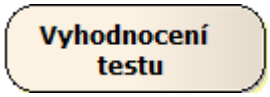
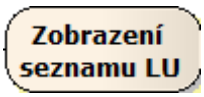
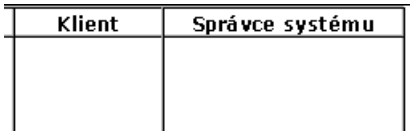
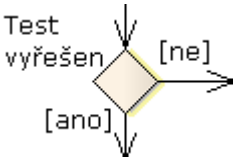
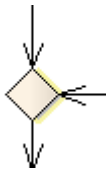
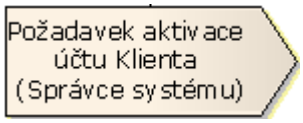
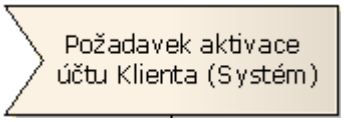
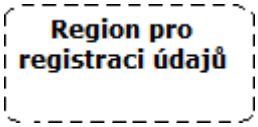
Neméně důležité je v diagramu aktivit určovat informace mezi sousedícími aktivitami konkrétním názvem, což je ve fázi implementace realizováno vstup/výstupními parametry realizovaného komponenty.

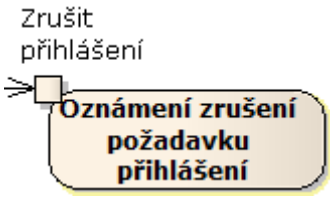
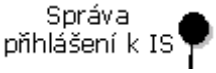
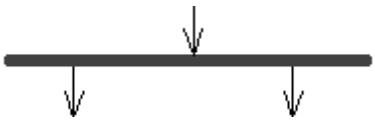
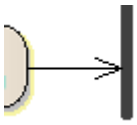
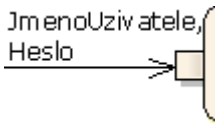
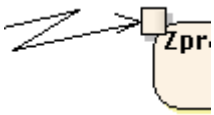


Obrázek 3.10 – Ukázka návrhu diagramu aktivit

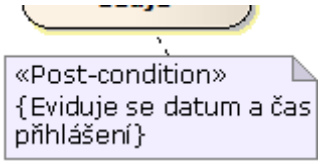
Diagram aktivit lze připojit k libovolnému modelovanému prvku, tj. konkrétnímu případu užití, třídě, rozhraní, komponentě. Diagramy aktivit využívají různé prvky, které jsou souhrnně uvedeny a popsány včetně jejich grafického symbolu v tab. 3.4.

Tab. 3.4 – Grafická syntaxe a popis prvků diagramu komponent

Prvky diagramu aktivit	Popis
Aktivita (Activity) 	Aktivita je určena k modelování podřízeného systému z důvodu zjednodušení hlavního diagramu aktivit nebo k využití jiného diagramu aktivit, již dříve definovaného toku událostí.
Akce (Action) 	Akce popisuje základní proces nebo transformaci, která vzniká mimo systém. Je základní a dále nedělitelným funkčním prvkem diagramu aktivit. Akce může být vykonávána buď člověkem v určité roli nebo systémem.
Oddíl (Partition) 	Logická organizace částí diagramu aktivit ohraničenými oblastmi, v nichž všechny aktivity patří účastníkovi, který je uveden v názvu dané oblasti. Neovlivňuje to řídicí tok diagramu aktivit, ale pomáhá zpřehlednění diagramu aktivit a zvýraznění přechodu řídicího toku mezi účastníky. Oddíly mohou být modelovány vertikálně i horizontálně.
Uzel rozhodování (Decision) 	V diagramu aktivit se pro účely řízení směru řídicího toku využívá prvek označený jako uzel rozhodování. Uzel rozhodování má standardně jeden vstup a dva výstupy. Směr toku závisí na výsledku podmínky, která bezprostředně patří k danému uzlu rozhodování.
Uzel sloučení (Merge Node) 	Sloučení různých řídicích toků, jejichž další tok řízení má procházet stejnými akcemi nebo aktivitami, tj. má společnou cestu, se provádí prvkem uzel sloučení. První aktivita, která následuje za tímto uzlem sloučení, očekává ukončení všech předchozích aktivit z obou slučovaných toků, pak teprve může být provedena následná činnost v diagramu aktivit.
Spuštění události (Send) 	Prvek generuje činnost spojenou se zasláním signálu. Signál může představovat např. písemný dopis zaslaný poštou, signál (hodnotu) poslanou s využitím hardwarové komunikace apod. Aktivita generuje fyzický signál, který je očekáván prvkem Příchozí událost.
Příchozí událost (Receive) 	Událost, která spouští následující řídicí tok po té, co obdržela požadovaný signál. Využívá se při modelování procesů, které čekají na příchozí signál (časový, hmotný). Po té pokračuje řídicí tok a další následující prvky diagramu aktivit. Může také souviset s prvkem Spuštění události nebo lze využít pouze samostatně.
Region (Region) 	Region je oblast zpracování toku informací. Jsou dvě základní skupiny regionu: pro činnost přerušení nebo pro vyjádření aktivit ve formě paralelního nebo opakovaného provádění některých aktivit/akcí.

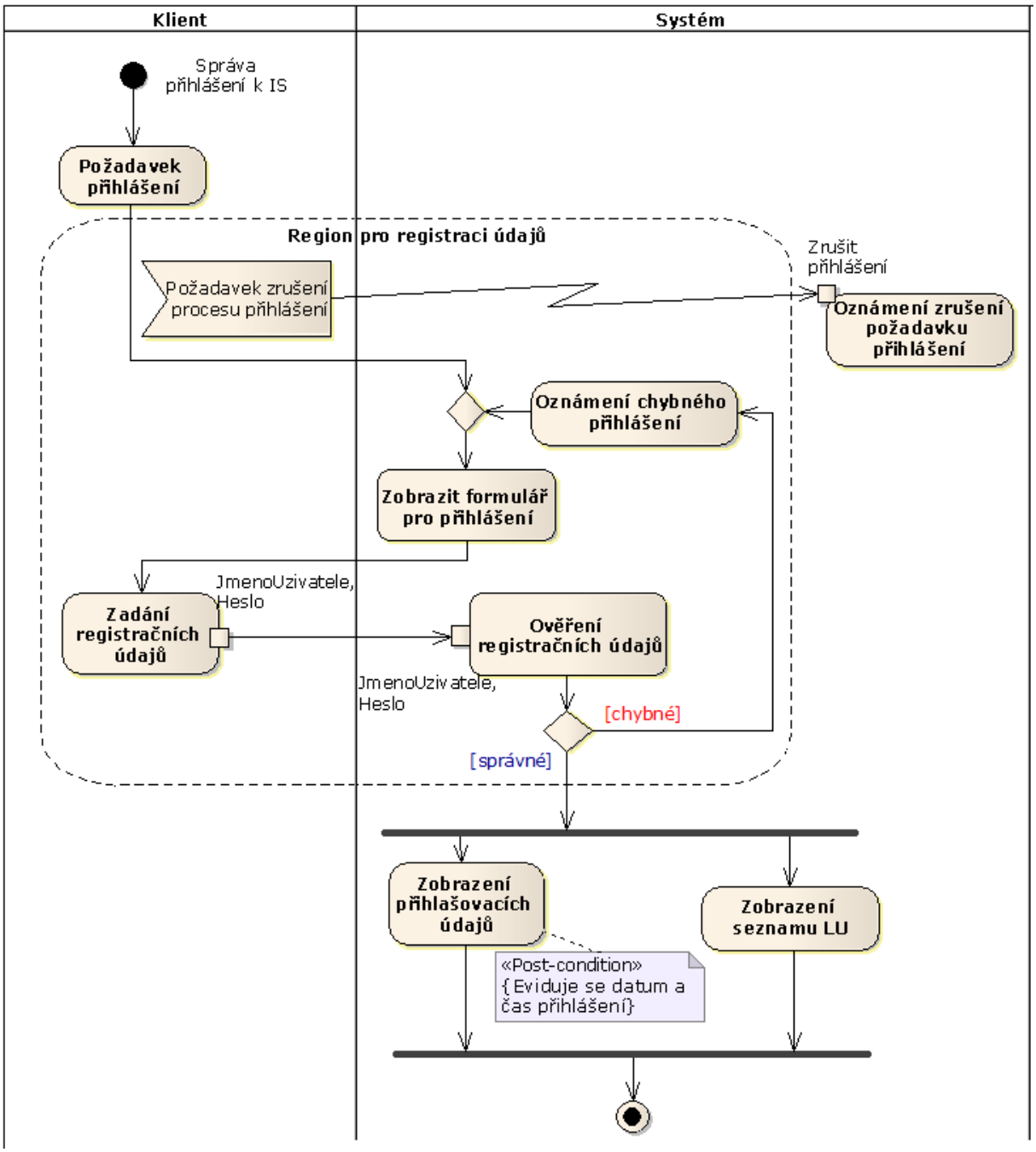
Zpracování výjimek (Exception) 	Prvek Zpracování výjimek je určen pro skupinu činností, které nastanou pouze v případě, když nastane neočekávaná událost. Je možné u daného portu tohoto prvku označit příchozí signál Zrušit přihlášení, který specifikuje signál neočekávané události. Název prvku označuje činnost zpracovanou pouze ve výjimečných případech. Pokud je potřebné ukončit celý proces aktivit (diagram aktivit), pak se za událost Zpracování výjimek vkládá prvek Konec řídicího toku.
Počáteční uzel (Initial) 	Aktivita začíná obvykle v bodě, který je zobrazen jako počáteční uzel symbolem kruhu (černé). Počáteční uzel definuje počátek řídicího toku diagramu aktivit.
Koncový uzel (Final Node) 	Diagram aktivit je modelem procesu, jehož činnost je konečná. Proto každá činnost je ukončena prvkem nazvaným koncový uzel. Pokud nenastane nepředvídaná situace, je diagram aktivit ukončen koncovým uzlem. Může také nastat situace, kdy se tok řízení dostane na „slepou cestu“, která zpravidla bývá ukončena jiným typem prvku ukončení, jehož popis následuje.
Konec řídicího toku (Flow Final) 	V diagramu aktivit jsou dva prvky, které ukončují řídicí tok diagramu aktivit. Koncový uzel je předpokládaným ukončovacím prvkem diagramu. Prvek konec řídicího toku se vkládá tam, kde modelovaný proces je zapotřebí ukončit.
Rozdělení/Spojení (Fork/Join) 	Rozdělení řídicího toku do několika paralelních toků s různými akcemi, které se provádějí nezávisle na sobě. Při použití prvku Spojení se synchronizují řídicích toků, tj. očekává se ukončení všech předchozích aktivit ze spojovaných toků.
Řídicí tok (Control Flow) 	Konektor spojující prvky nebo uzly v diagramu aktivit. Zobrazuje se plnou čarou ukončenou šipkou, která označuje směr řídicího toku.
Tok objektu (Object Flow) 	Tok objektu spojuje dva objekty s graficky vyznačeným portu na obou stranách prvků diagramu aktivit. Je určen výhradně k předávání informací mezi dvěma činnostmi, které jsou určeny názvem dané informace na obou stranách této vazby. Tím je přesně dáno, jaké informace jsou předávány a taky potřebné pro další činnost.
Tok přerušení (Interrupt Flow) 	Tok přerušení je pojení, které se aktivuje pouze v neočekávaných stavech. Pokud nastane neočekávaná situace, je splněna aktivity, která je spojena tokem přerušení s prvkem případě přerušení způsobené splněním aktivity od níž přerušení přichází.

Omezení akce ()



Řídicí tok lze podmínit pomocí dvou typů omezení: vstupních (pre-condition) a výstupních (post-condition). Omezení je vázáno na konkrétní akci. V případě vstupní podmínky se akce spustí až po jejím splnění. Pokud se jedná o podmínku výstupní, tak se provede daná akce, pak se čeká na splnění výstupní podmínky a v případě, že je splněna, je předáno řízení další akci nebo prvku, který bezprostředně následuje.

Následnost dílčích činností v diagramu aktivit je dána uskutečněním dílčích následných aktivit bezprostředně následujících za sebou. Pokud chcete zdůraznit, že mezi aktivitami předáváte signály, bez nichž nelze další činnost provést, pak využijte prvky Spuštění události a Příchozí událost.

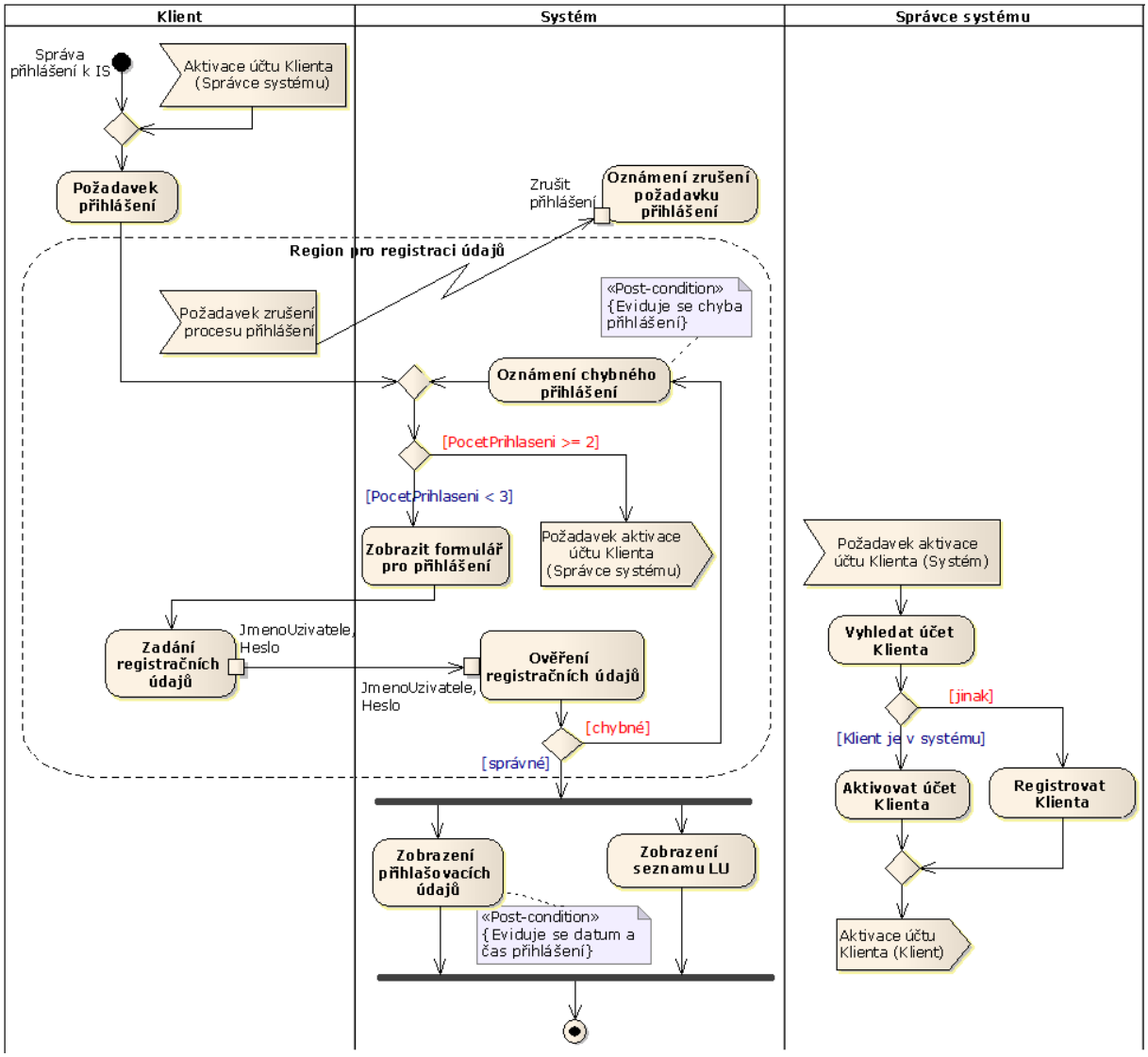


Obrázek 3.11 – Diagram aktivit správy přihlášení k IS bez omezení počtu přihlášení

Pokud v diagramu aktivit existuje více činností, které se mohou nezávisle na sobě provádět, snažte se využít prvku Rozdělení/Spojení tak, aby bylo znázorněno lépe paralelní zpracování činností více účastníky a zdůrazněna tak další skupina činností, které vyžadují provedení těch předchozích. Pro jednoho účastníka není doporučeno využívat při modelování prvku Rozdělení nebo Spojení.

Řešený příklad

Jedním z diagramů aktivit navržených pro modelovaný systém je zaměřen na objasnění činností při procesu přihlašování do IS. Takto vznikla nejprve jednodušší verze diagramu aktivit bez omezení počtu chybných přihlášení (obrázek 3.11). Posléze byl proces přihlášení rozšířen o omezení počtu přihlášení a také zahrnutí činnosti aktivace účtu *Klienta*, kterému byl z jistých důvodů zamezen přístup k IS.



Obrázek 3.12 – Diagram aktivit správy přihlášení k IS s omezením počtu přihlášení

Celkově byl diagram aktivit koncipován do vertikálního směru oddílů se dvěma účastníky *Klient* a *Systém*. Činnosti jsou typické pro běžné registrační procesy. Není opomenuta možnost zrušení přihlášení z jakýchkoliv důvodů registrujícího se účastníka s využitím prvku *Region pro přihlášení*. V rámci přihlašování jsou kromě řídicích toků využity také objektové toky přenášející informace se jménem a heslem přihlašovaného uživatele IS. Následuje kontrola registračních údajů a zobrazení informací v úvodní

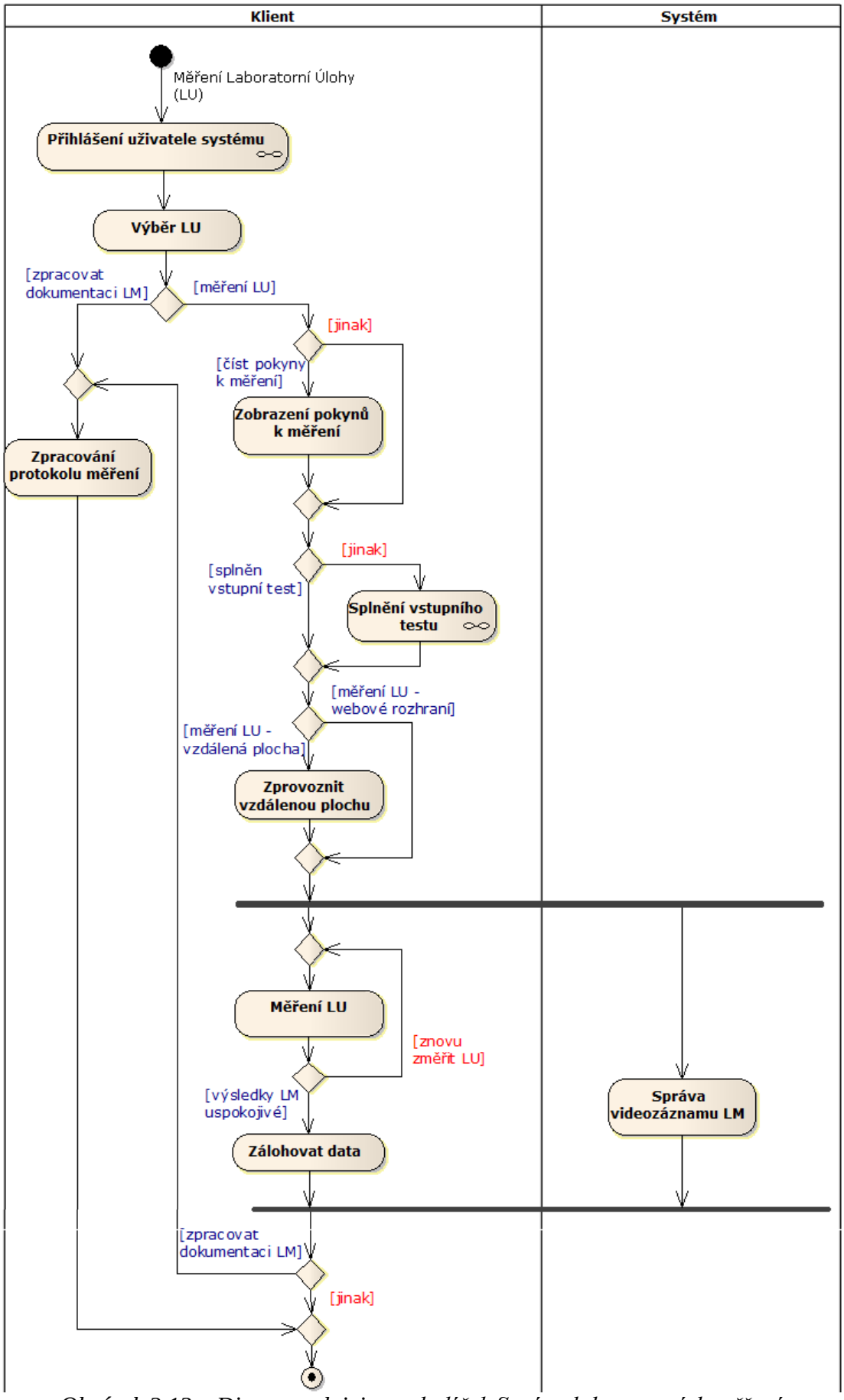
obrazovce. Podmínka <<*Post-condition*>> vyjadřuje pro implementaci další činnosti, které souvisí s aktivitou **Zobrazení přihlašovacích údajů** bezprostředně následujících po ukončení této aktivity.

V rozšířené verzi činnosti **Správa přihlášení k IS** v diagramu aktivit (obrázek 3.12) je omezen počet přihlášení uživatele a poté je účet *Klienta* zablokován. V případě, že účastník *Klient* zjistí v průběhu procesu přihlášení, že zapomněl heslo nebo registrační údaje, pak přerušením činnosti prostřednictvím prvku *Příchozí událost* s konkrétním názvem **Požadavek zrušení procesu přihlášení** se spustí následná činnost prostřednictvím toku přerušení **Oznámení zrušení požadavku přihlášení**, kde se *Klient* vrátí zpět na začátek IS, kde je možné požádat o aktivaci účtu (pro získání nových přihlašovacích údajů). Registraci zde *Klienti* sami neprovádí, neboť jsou v rozhodném období zavedeni automaticky *Správcem systému*. Tento informační systém je určen jen pro předem známou skupinu uživatelů a nemůže přistupovat kdokoliv, kdo se zaregistruje.

Diagramy aktivit (obrázek 3.11 a obrázek 3.12) pro **Správu přihlášení k IS** byly vytvořeny na základě požadavku programátora IS, který požadoval bližší objasnění činnosti definované v předloženém scénáři případu užití. Vždy při analýze vycházíme z pravidla tvorby minimálního počtu diagramů dle požadavků členů realizačního týmu, která nastane při dialogu nad předloženou analýzou.

Další případovou studií analýzy je diagram aktivit objasňující návaznost případů užití z balíčku **Správa laboratorních měření**. Diagram případů užití je zpravidla navržen tak, aby případy užití, které jsou výše položené, byly provedeny dříve při zpracování činnosti v čase. Podmínky, kdy má být případ užití vykonán, jsou uvedeny v popisu scénáře. Nicméně příliš podmíněná činnost zpracování jednotlivých případů užití v rámci daného balíčku není moc přehledná. Diagram aktivit (obrázek 3.13) přehledně popisuje návaznost činností a podmínek, které musí být splněny pro činnost **Měření laboratorní úlohy**. Paralelní činnost **Správy videozáznamu LM** není možné vyjádřit v rámci scénáře, ale pouze v diagramu aktivit.

Poslední aktivitou v rámci **Měření laboratorní úlohy** je činnost **Zpracování protokolu měření**. Údaje o zpracování úlohy nebudou *Klienti* provádět formou textové zprávy, ale předem strukturovaného dokumentu, který bude naplněn s využitím IS a jeho striktně definovaných částí dokumentace. Přístup ke **Zpracování protokolu měření** bude tedy pouze po přihlášení do IS, naměření potřebných dat. Pak teprve bude *Klient* přistupovat k poslední činnosti **Zpracování protokolu měření**. Cílem řízené dokumentace je zamezit kopiím informací mezi jinými *Klienty* IS, podpořit jejich samostatnou činnost a také přípravu před měření. Zda se záměr podaří, ukážou časem zkušenosti z provozu IS.



Obrázek 3.13 – Diagram aktivit pro balíček Správa laboratorních měření

Shrnutí pojmů

Analýza modelovaného systému spočívá v přesně definovaných činnostech daného systému. Proto je nutné provést popis dílčích činnosti velmi důsledně, aby nemohly nastat žádné chyby v implementaci. Proto je důležité se zamyslet, zda je postačující pro potřeby sestavené analýzy souhrn scénářů k daným případům užití nebo je nutné více specifikovat a lépe znázornit vnitřní strukturu problému. K tomu je možné využít sekvenčních diagramů nebo diagramů komunikací.

V podstatě je sekvenční diagram a diagram komunikací shodným nástrojem pro potřeby analýzy. Rozdíl mohou uživatelé spatřit pouze v samotné podobě diagramů. Sekvenční diagramy jsou pro časovou následnost aktivit názornější. Diagramy komunikací jsou kompaktnější a možná názornější z pohledu souhrnu účastníků, objektů a tříd v rámci všech aktivit.

V této kapitole je popsán ještě diagram aktivit, který není specificky určen pouze pro znázornění činností náhradou textového popisu scénářů případů užití, ale zpravidla se užívá pro vyjasnění vazeb a souvislostí mezi dílčími případy užití nebo také mezi případy užití z různých balíčků. Diagramy aktivit nabízí jedinečnou možnost modelování paralelních toků činností, které probíhají současně.

Otázky

1. Diagram komunikací patří do skupiny strukturních diagramů nebo diagramů chování a interakcí?
2. V jaké situaci využijete spíše sekvenční diagram než scénář pro podrobnou analýzu konkrétního případu užití?
3. Je nezbytné sestavovat sekvenční diagramy v analýze systému pro každý případ užití?
4. Popište způsob, jakým se vyznačují iterace (cykly) v rámci sekvenčního diagramu, diagramu komunikací a diagramu aktivit.
5. Vysvětlete pojem „pre-condition“, „post-condition“. V jakých diagramech je analytik aplikuje?
6. Může být spojen účastník s entitní třídou přímou vazbou v sekvenčním diagramu?

CD-ROM

Analýza informačního systému je realizována v produktu Enterprise Architect, verze 8 firmy Sparx Systems Pty Ltd. Animační ukázky pro tuto kapitolu zahrnují důležité části a postřehy při **návrhu sekvenčního diagramu** a **diagramu aktivit**. Animace jsou zaznamenány programem Adobe Captivate 4.0.

4 OBJEKTOVÉ MODELOVÁNÍ NEBOLI ANALÝZA A NÁVRH

Čas ke studiu: 2 hodiny

Cíl: Po prostudování tohoto odstavce budete umět

- Modelovat business objekty.
- Definovat pojmy z oblasti objektového/datového modelování.
- Analyzovat a navrhovat modely tříd.
- Analyzovat a navrhovat datové modely.
- Řešit jednoduché úlohy z oblasti modelování.

Výklad

Pojmem objekt z problémové domény je označován **business objekt** a odpovídá libovolnému předmětu, osobě v reálném prostředí modelované oblasti. V procesu modelování jsou business objekty transformovány postupně na **softwarové objekty**.

Softwarové objekty se v oblasti objektového modelování označují třídy a jsou určeny pro implementaci v prostředí C#, C++, Java, Visual Basic, Python, .NET a dalších. Pokud je implementace určena pro databázové prostředí, pak se softwarové objekty v oblasti databázového modelování nazývají relační tabulky. Proces objektového modelování v UML je obecně zaměřen na modelování diagramů tříd, proto je následující text takto směrován. V kapitole 0 je pak uveden postup mapování objektového modelu na datový model.

Softwarový objekt zahrnuje chování a stav modelovaného business objektu.

- **Stav softwarového objektu** – představuje aktuální kombinaci hodnot všech jeho atributů.
- **Chování softwarového objektu** – představuje nabídku činností, které lze s objektem provádět.

Všechny business objekty mají v problémové oblasti svou identitu, jsou nezaměnitelnou jednotkou. V případě, že existuje několik softwarových objektů, které mají stejné stavy a chování pouze s jinými konkrétními hodnotami stavů, pak je možné označit tyto objekty jako **instance tříd**. Pojem **třída** pak představuje abstraktní objekt pro skupinu softwarových objektů reálně existujících v modelovaném systému obsahující stejné **atributy** (označení pro stavy) a **operace** (označení pro chování). **Model tříd** je obecnějším modelem než **datový model**, který je určen speciálně pro implementaci business objektů v prostředí relačních databází.

Z hlediska životního cyklu se objektové modelování dělí do tří úrovní:

Konceptuální (datový) model – popisuje model sestavený z entit, relací a je obecně nastaven tak, že *je nezávislý na implementačním prostředí*.

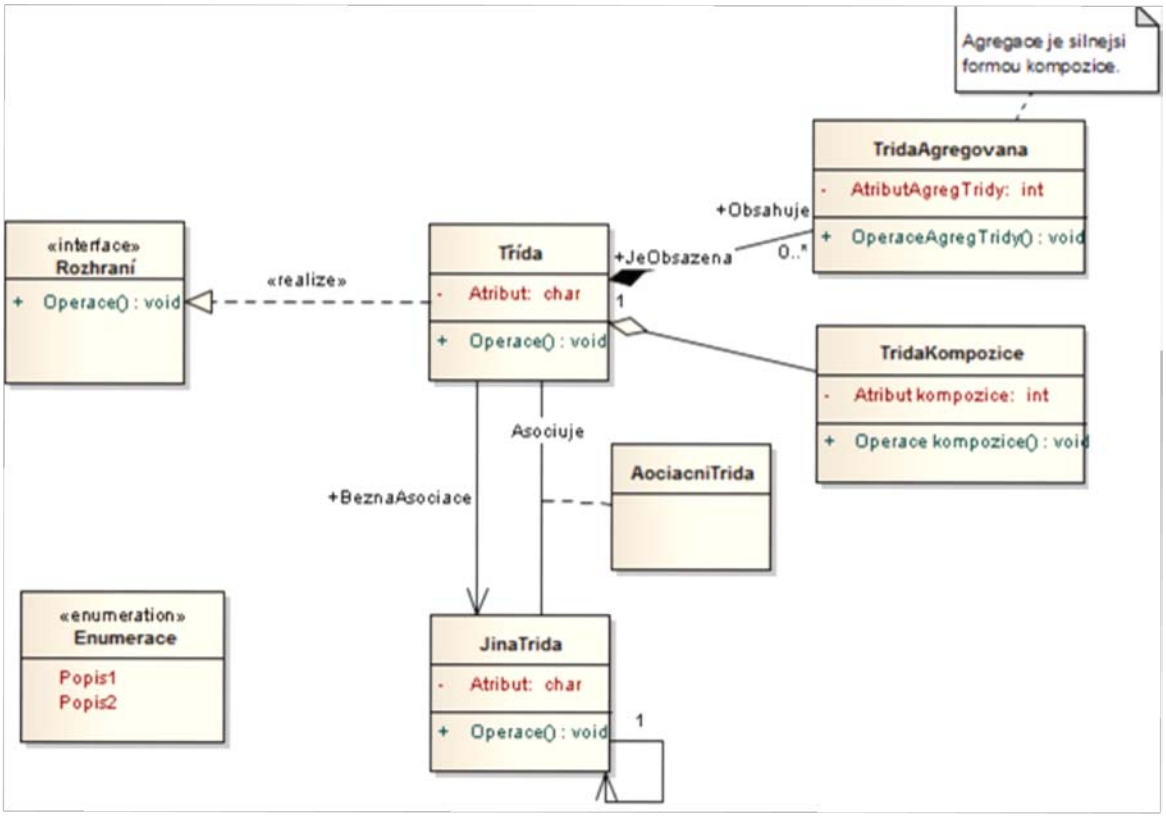
Logický model – definuje konceptuální schéma z pohledu databázové implementace (tabulky, třídy nebo XML tagy dle zvoleného implementačního prostředí).

Fyzický model – je nejkonkrétnější schéma popisující konkrétní soubory fyzických dat a jednotlivých atributů. Fyzický model představuje diagram nasazení (kapitola 5.2).

Nejprve začneme popisem návrhu konceptuálního modelu (diagramu tříd) a podrobným popisem charakteristik objektového modelování ve formě konceptuálního modelu. Logický model je dále představen prostřednictvím datového modelu pro databázovou implementaci modelovaného systému. Datový model vzniká mapováním diagramu tříd do prostředí databáze, ale může být také navržen přímo bez vzniku diagramu tříd. Pak takové objektové modelování nebude mít konceptuální schéma, ale pouze logický a fyzický model.

4.1 Diagram tříd

Diagram tříd představuje primární nástroj pro modelování business objektů, jejich vlastností a vztahů v prostředí modelovacího nástroje UML, je základním diagramem pro generování kódu. Všechny ostatní diagramy čerpají informace z diagramu tříd. Diagram tříd se skládá ze dvou základních entit: **třída** a **relace**.



Obrázek 4.1 – Diagram tříd (obecné vyjádření vazeb, tříd, atributů a operací).

Kromě těchto základních entit diagram tříd může obsahovat také prvky známé z jiných diagramů např. *balíček* a *rozhraní* nebo také specifické druhy těchto základních entit, tj. *tabulka* je specifikací entity třída (má konkrétní implementační podobu).

Třída

Pojem třída je objekt, který vzniká přechodem z business objektu na konceptuální objekt a jeho dalším upřesněním. Někdy je možné se také setkat s pojmem softwarový objekt, což představuje v softwarovém prostředí entitu s názvem **třída**. V diagramu tříd je symbol třídy definován pomocí tří základních částí:

- název (definice) třídy,
- seznam atributů dané třídy,
- seznam operací dané třídy.

Jednotlivé části mají svou pevně stanovenou syntaxi, jejíž části jsou, buď povinné, nebo nepovinné (v hranatých závorkách). Mezitím jsou také povinně vkládané symboly této syntaxe, např. symbol „:“ nebo „=“. Následující část této podkapitoly je věnována deklaraci zápisu třídy, konkrétně všech jejích částí atributů, operací a jejich parametrů.

Syntaxe atributu

[přístup] název: [typ násobnost] [hodnota] [příznaky]

přístup	Označuje viditelnost atributu: + (public), - (private), ~ (package), # (protected).
název	Řetězec jednoznačně definující atribut.
typ	Určuje datový typ proměnné.
násobnost	Určuje počet objektů, které mohou být v atributu umístěny (1, 0..1, 5..8, 1..*, 0..* a další)
hodnota	Stanoví výchozí hodnotu atributu.
příznaky	Vyjadřují další vlastnosti atributu. (read-only, ordered a jiné)

Syntaxe operace

[přístup] název ([seznam parametrů]): [typ] [příznaky]

přístup	Označuje viditelnost operace: + (public), - (private), ~ (package), # (protected).
název	Řetězec jednoznačně definující operaci.
seznam parametrů	Definuje seznam parametrů operace.
typ	Stanoví datový typ návratové hodnoty operace.
příznaky	Vyjadřují další vlastnosti operace. (read-only, query a jiné)

Seznam parametrů

název: typ = hodnota

název	Řetězec jednoznačně definující název parametru.
typ	Určuje datový typ parametru.
hodnota	Stanoví výchozí hodnotu parametru.

Poznámka k syntaxi atributů, operací a seznamu parametrů

Položky v hranatých závorkách jsou nepovinné. Černou barvou jsou vyznačeny oddělovače, které jsou nedílnou součástí syntaxe. Pořadí prvků v syntaxi musí být dodrženo.

Relace diagramu tříd

Asociace představují obecný vztah mezi dvěma třídami softwarových objektů. Definuje, jak navzájem k sobě mohou přistupovat a co je přípustné. Stejně jako je důležité správně označit názvy tříd, atributů a operací, je také důležité správně definovat typ a název asociace. Základní typy asociací:

- asociace (jednosměrná/obousměrná),
- agregace,
- kompozice,
- generalizace.

Seznam základních druhů asociací, popisu jejich aplikace v diagramu tříd včetně jednoduchých příkladů vázaných tematicky na modelovaný systém z předchozích příkladů je souhrnně uveden v následující tabulce Tab. 4.1.

Násobnost asociací

Pojem násobnost asociace se určuje podle definované role k dané asociaci. Pak podle počtu objektů dané třídy, které mohou navazovat na konkrétní objekt jiné třídy spojené s ní asociací, je definován typ násobnosti, např.:

1 právě jeden,

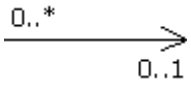
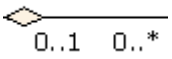
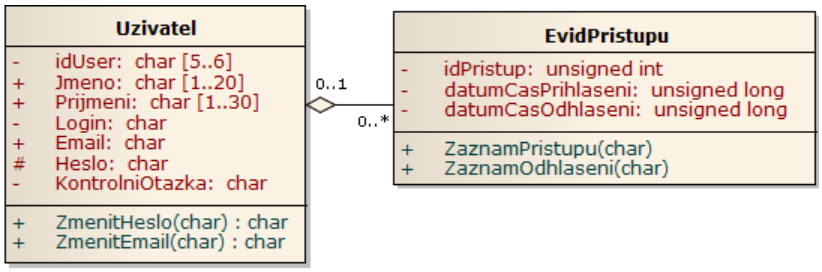
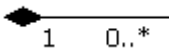
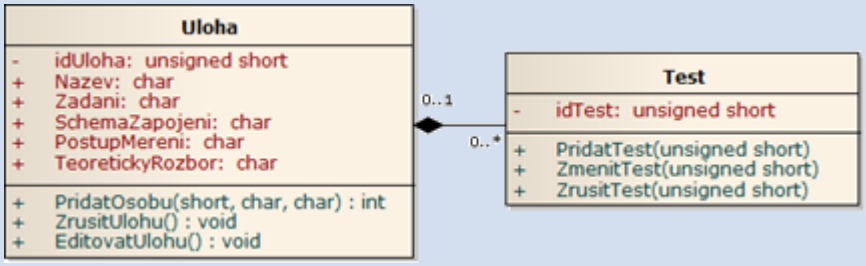
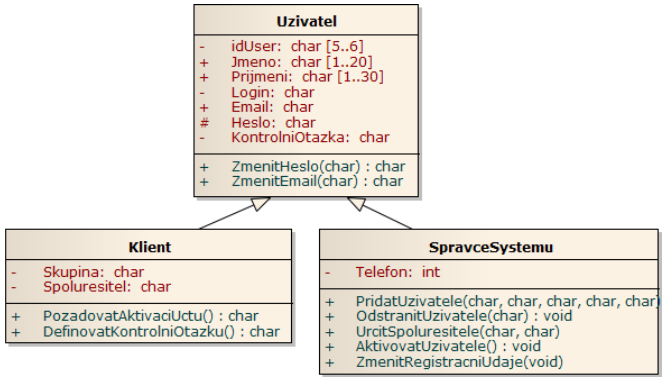
0..1 žádný nebo jeden,

0..* žádný nebo jeden nebo více,

1..* jeden nebo více.

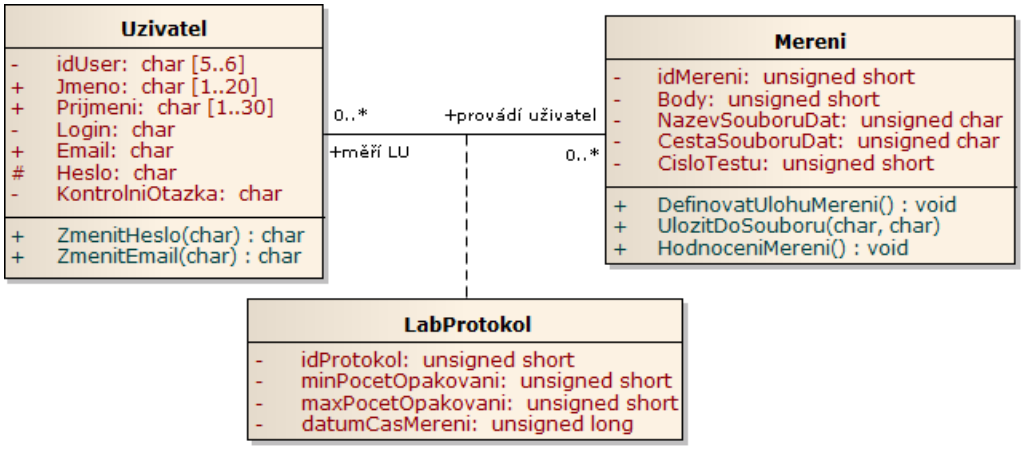
Ve většině případů se využívají výše uvedené označení pro násobnost asociací, ale pro konkrétní rozsah hodnot je možné uvést dolní a horní mez násobnosti asociace, které vyřeší problém s další omezující specifikací dané asociace.

Tab. 4.1 – Grafická syntaxe a popis prvků diagramu komponent

Relace diagramu tříd	Popis
Asociace (Associate) 	Asociace indikuje vztah mezi dvěma třídami, který se v diagramech UML přesně definuje. Asociace je graficky zobrazena plnou čarou a jednoduchou šipkou na konci asociace. Asociace jsou jednosměrné nebo obousměrné. Každé asociaci je vhodné definovat její roli (slovní popis vztahu mezi třídami). Podle konkrétní role a dané asociace je důležité také označit její násobnost.
Agregace (Aggregate) 	Agregace je typem asociace, která přiřazuje k nadřazené třídě další třídy, které mohou být její součástí. Součásti, ve významu agregace, nemusí být vždy spojeny s nadřazeným objektem. 
Kompozice (Compose) 	Kompozice je typem asociace, která definuje nadřazenou třídu a její nezbytné součásti, které k ní patří. Součásti nemohou samostatně existovat bez jejich nadřazené třídy (pro konkrétní případ je lépe se vyjadřovat o instanci). 
Generalizace (Generalize) 	Vztah generalizace (zobecnění) využívá vlastnosti dědičnost. Existuje-li několik tříd, které mají společné vlastnosti, je možné vytvořit společnou (zobecněnou) třídu, která bude zahrnovat společné prvky (atributy, operace) a její potomci pak budou definovat pouze své specifické vlastnosti. 

Asociační třídy

Vznik asociačních tříd je dán výskytem informací (atributů), které popisují danou asociaci mezi dvěma třídami, a tyto informace nelze přiřadit k žádné z tříd této asociace. Zpravidla vznikne asociační třída standardně mezi násobnou asociací na obou stranách asociace. Pak není možné přiřadit atributy, které souvisí s konkrétní instancí vazby, do žádné z asociovaných tříd.



Obrázek 4.2 – Asociační třída v diagramu tříd

Vznik asociační třídy je dán podmínkou náležitosti každé instance asociační třídy k právě jediné instanci třídy na levé i pravé straně asociace. Pro analýzu vzniku asociační třídy je vhodné provést rozbor několika možných instancí tříd a konkrétních atributů, které patří k jednotlivým asociačním vazbám.

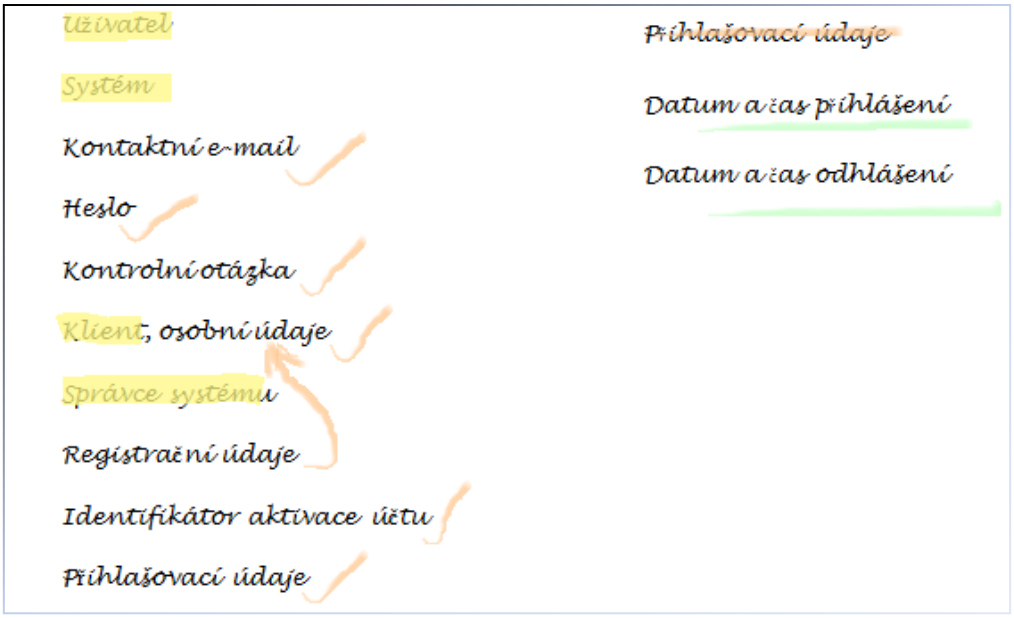
Řešený příklad

Analýza a návrh modelu tříd vychází zpravidla z diagramu případů užití, z nichž analytik vyhledává a navrhuje třídy pro skupiny společných vlastností, které může sdružovat do společných tříd a přiřadit jim společné atributy a operace.

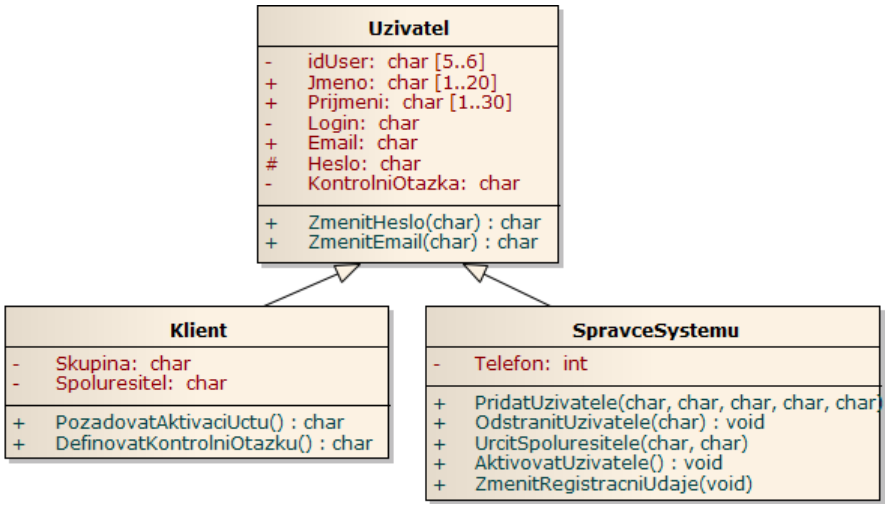
V příkladu řešeném v tomto dokumentu byly v diagramu případu užití navrženy čtyři balíčky (**Správa uživatelů**, **Správa laboratorních měření**, **Zpracování dokumentace**, **Archivace dat**). Při analýze balíčku **Správa uživatelů** je vhodné provést rozbor všech případů užití a jejich scénářů, respektive sekvenčních diagramů. Analytik se zaměří zpravidla na výrazy dané podstatnými jmény v popisu, které mohou v diagramu tříd představovat název třídy, případně název atributu. Je tedy nutné analyzovat souvislosti mezi pojmy a navrhovat třídy.

V balíčku **Správa uživatelů** tedy z případů užití můžeme analyzovat pojmy *Uživatel*, *Klient*, *Systém*, *Správce systému*, což jsou navržení účastníci systému, což samo o sobě neznamená, že pro tyto skupiny navrhneme dílčí třídy. Třídu navrhuje analytik až tehdy, když informace (atributy) daného pojmu je nutné v navrhovaném systému uchovávat. K atributům pak IS přistupuje prostřednictvím operací, které získávají obsah atributu nebo jej naopak změní. Z analýzy (Obrázek 4.3) tedy byly zatím navrženy třídy **Klient**, **SprávceSystému** a

Uživatel (třída zobecněná) s atributy (označeny oranžovým zatržením). Položky podtržené zeleně se pro každého konkrétního *Klienta* v rámci systému mění, proto budou evidovány v samostatné třídě **EvidPřístupu**.



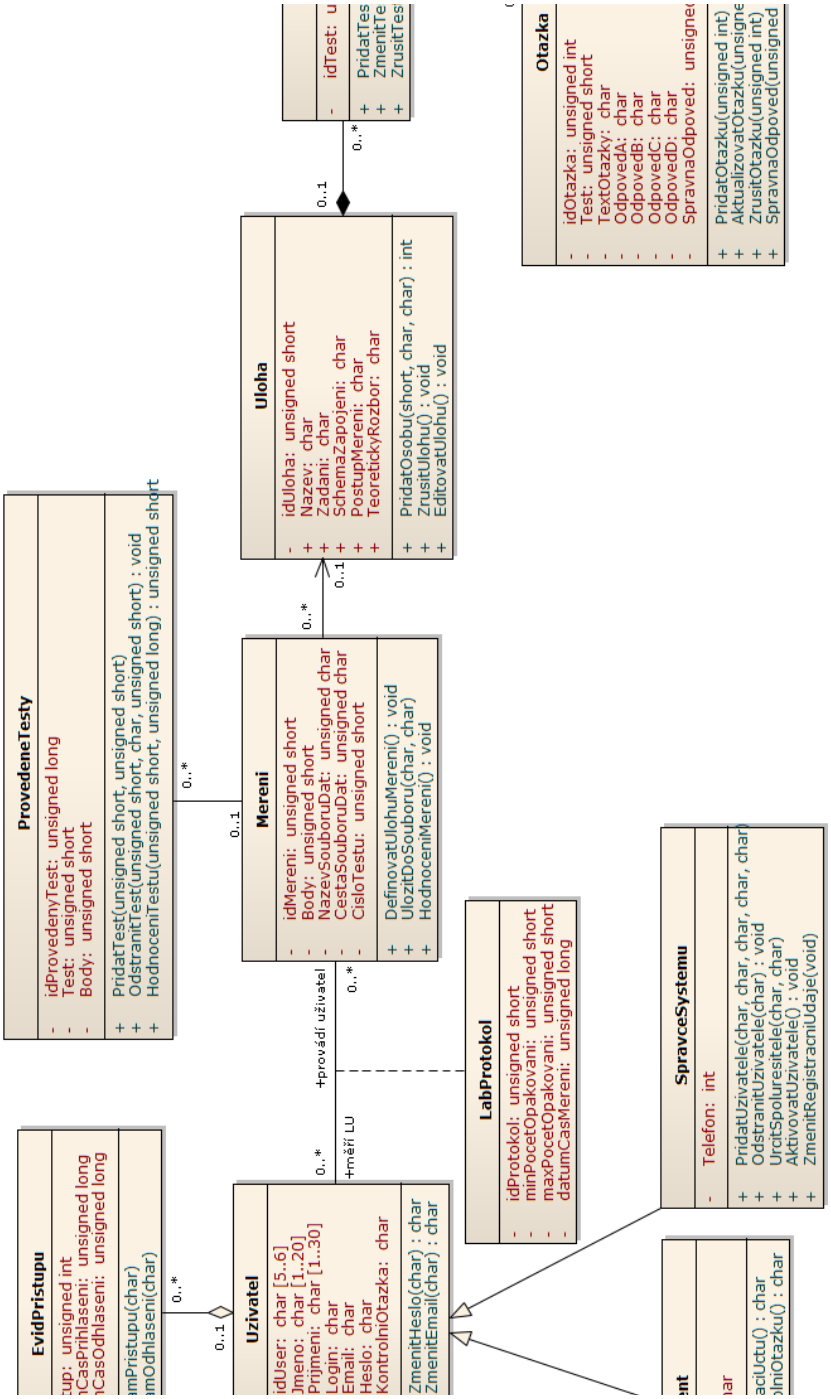
Obrázek 4.3 – Analýza balíčku Správa uživatelů (návrh tříd a atributů)



Obrázek 4.4 – Analýza balíčku Správa uživatelů (návrh tříd, atributů a asociací)

Vzhledem k různorodosti dalších analyzovaných atributů a dalších operací byly navrženy třídy samostatně pro účastníky *Klient* a účastníka *Správce systému*, jejich operace jsou rozdílné (Obrázek 4.4) s generalizační vazbou na třídu **Uživatel** obsahující společné atributy a společné operace, které využívají také třídy **Klient** a **SprávceSystému**.

V počátku návrhu tříd vyvstane skupina tříd a jejich atributů a poté analyzujeme postupně asociace mezi třídami. V situacích, kdy si analytik není jist typem nebo násobností asociace, vzniká nejprve návrhová studie instancí neboli konkrétních objektů modelovaného systému, z nichž pak lépe analytik navrhuje správnou asociaci a její násobnost.



Obrázek 4.5 – Analýza systému Evidence měření

Celkový popis a postup návrhu tříd už dále popsán nebude a to také z důvodu implementačního prostředí, které vyžaduje v etapě analýzy a návrhu datový model pro implementaci modelovaného systému v databázi SQL Serveru. Náhled na kompletní diagram tříd pro modelovaný systém Evidence měření je na obrázek 4.5.

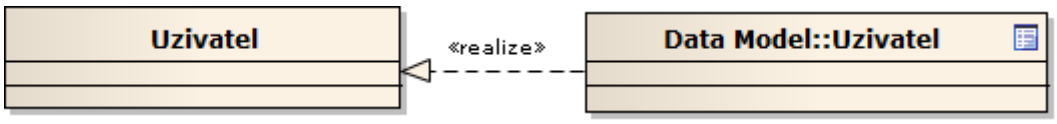
Výklad

4.2 Datový model

Datový model vzniká mapováním objektového modelu (modelu tříd) do prostředí relačních databází. Následující postup mapování by měl být spíše inspirací než striktním návodem z důvodu jeho individuality vzhledem k modelované problematice.

Mapování tříd do tabulek

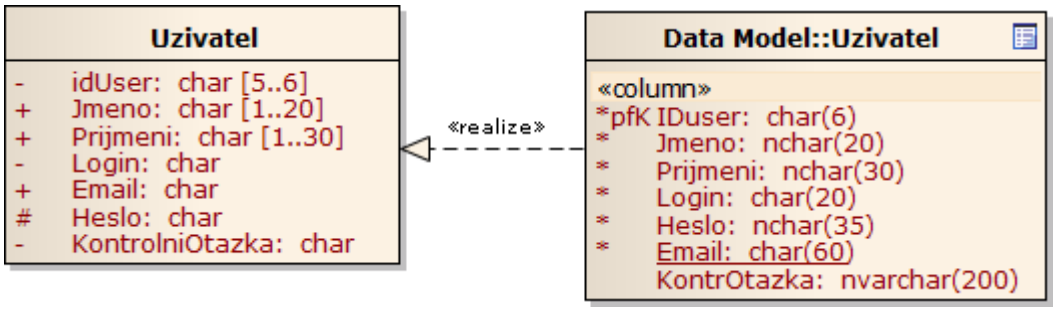
Tabulka v datovém modelu vzniká mapováním třídy objektového modelu, která je vyznačena stereotypem <<table>>.



Obrázek 4.6 – Mapování tříd do tabulek

Mapování atributů do sloupců

V procesu mapování atributů do sloupců se uvažují pouze perzistentní atributy tříd, tj. atributy, jejichž hodnoty nejsou odvozeny z hodnot jiných atributů. V souvislosti s mapováním atributů do sloupců dochází také k mapování datových typů atributů do datových typů sloupců s ohledem na konkrétní implementační prostředí.



Obrázek 4.7 – Mapování atributů do sloupců

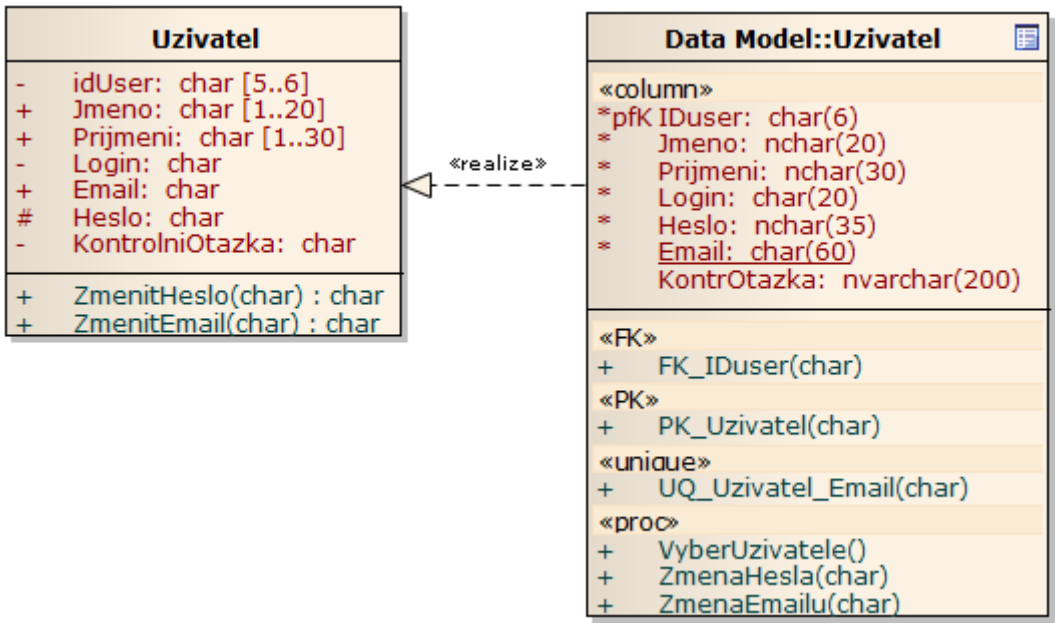
Atributy v datovém modelu mají své specifické vlastnosti, které budou podrobně vysvětleny v následující podkapitole. Graficky jsou v objektu tabulka vyznačeny například znaky „pfK“ nebo hvězdičkou před názvem atributu nebo jeho podtržením čarou.

Mapování operací do objektů modelujících chování entity

Objektový model popisuje své chování seznamem operací přidružených k dané třídě. Mapování chování objektu do datového modelu představuje přiřazení business nebo logických pravidel k dané entitě prostřednictvím následující skupiny objektů:

Primární klíč (Primary Key Constraint, PK) – jednoznačně identifikuje každou instanci dané entity (neboli řádek tabulky). Primární klíč je dán pouze jedním atributem a to buď takovým, který zajistí jednoznačnost daných instancí nebo nově definovaným atributem generovaným jako globálně jedinečný identifikátor (GUID, globally unique identifier).

Cizí klíč (Foreign Key Constraint, FK) - je atribut entity, který odpovídá primárnímu klíči jiné entity s ní spojené danou asociací. U násobné asociace nelze zvolit atribut, který je již primární klíčem dané entity, neboť jeho hodnoty mohou být pro různé instance (řádky tabulky) stejné.



Obrázek 4.8 – Mapování operací do objektů modelujících chování entity

Omezení UNIQUE (Uniqueness Constraint) – definuje seznam atributů, jejichž sloupce nesmí obsahovat duplicitní hodnoty. V případě, že vznikne požadavek na uložení duplicitní hodnoty, pak databáze nepřipustí její uložení.

Omezení CHECK (Validity Check) – definuje pravidlo, podle nějž dochází k ověření validity dat pro jejich následné uložení do databáze. Pravidlo definuje nejen množinu možných hodnot pro daný atribut, ale může také definovat pravidla pro validitu mezi ostatními atributy.

Index (Index Constraint) - představuje setříděný pohled na tabulku z důvodu zajištění vyššího výkonu zpracování výsledných informací nad danou tabulkou, např. při prohledávání hodnot.

Spoušť (Trigger) – představuje chování spojené s danou entitou pro zajištění integrity dat, kdy nepostačují výše uvedené typy omezení. Proces chování se spouští s činností dané entity, konkrétně při vložení/odstranění dat nebo před/po aktualizaci dat.

Uložená procedura (Stored Procedure) – představuje významné rozšíření funkcionality databáze tvorbou skriptu dle pravidel implementačního prostředí.

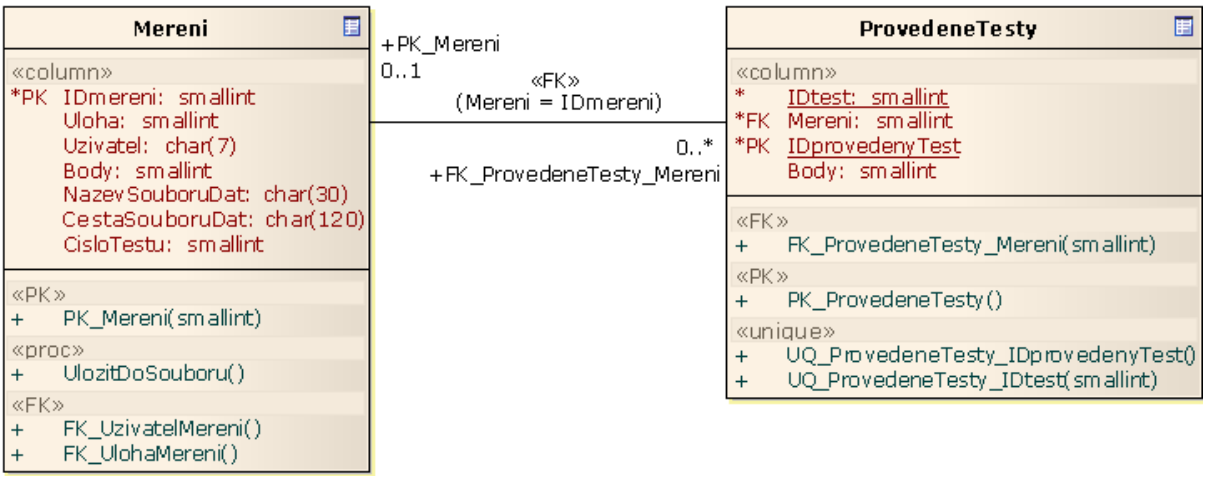
Mapování asociací mezi objekty

Asociace představuje strukturální vztah mezi objekty, který je mimo jiné vyjádřen také konkrétní rolí k objektu. Pro jednoznačné mapování asociací v datovém modelu je nutné v obou objektech zajistit existenci atributu, jehož datový typ i obsah je stejný, tudíž je nutné

u mapování asociací zajistit vznik nebo přiřazení vhodného typu atributu, který bude označen jako cizí klíč.

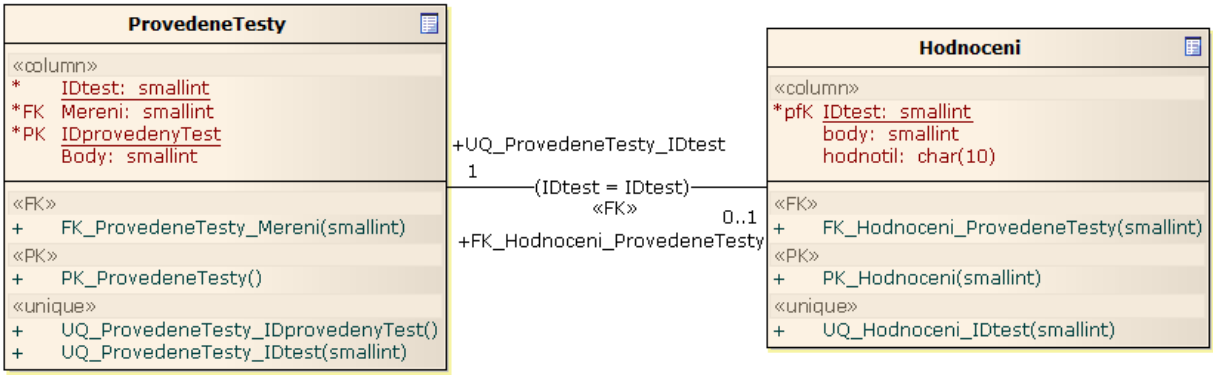
U asociací rozlišujeme dva základní pojmy, násobnost a směrovost. Násobnost asociace mezi dvěma objekty může nabývat těchto hodnot 1:1, 1:N (N:1), M:N.

U násobných asociací „One-to-many“ (1:N) nebo „Many-to-one“ (N:1) se mapuje asociace přiřazením primárního klíče (PK) atributu entity a vazbou jedinečnou (1) a **vytvořením nového atributu cizího klíče** (FK) u entity s vazbou mnohoznačnou (N). Atribut s označením cizího klíče obsahuje pouze takové hodnoty instancí, které je možné najít v entitě s primárním klíčem.



Obrázek 4.9 – Mapování násobné asociace 1:N (N:1)

Asociace „One-to-one“ (1:1) je pouze specifickým případem asociace 1:N (N:1). Její mapování probíhá stejným způsobem, včetně vytvoření cizího klíče pro jednu z entit. Obě entity obsahují stejné primární klíče se stejným datovým typem a asociace mezi entitami vzniká propojením atributů primárních klíčů. Násobnost asociace 1:1 je pro obě entity možné zvolit 1 nebo 0..1. Vznik asociace 1:1 je aplikován v datovém modelu např. z důvodu rozdělení atributů do dvou samostatných entit nebo při zařazení již existující entity z jiného modelovaného systému do stávajícího modelu. Nevýhodou těchto samostatných entit bude v další části implementace složitější zajištění operací nad oběma entitami.



Obrázek 4.10 – Mapování násobné asociace 1:1

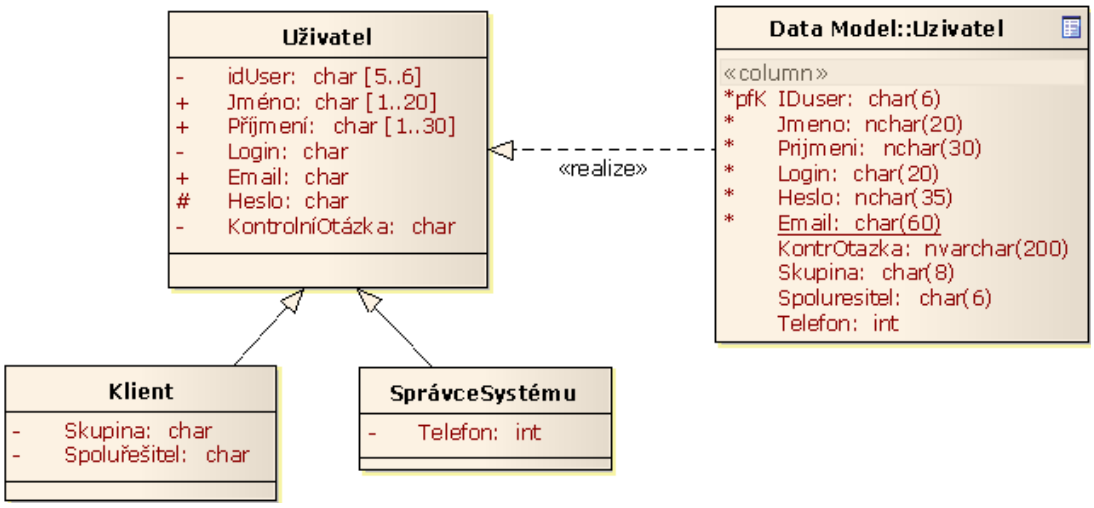
Jednosměrné a obousměrné asociace mezi třídami, respektive tabulkami mají stejnou násobnost i role uvedené na obou koncích asociace, tudíž žádná změna při procesu mapování těchto asociací nenastává.

Jednosměrná asociace je označena šipkou na jedné straně asociace. Tento směr šipky asociace udává „kdo koho ve vztahu asociace využívá“. V syntaxi UML je neoznačení směru asociace šipkou považováno za obousměrnou asociaci (šipka na obou stranách asociace se nevyužívá). V datovém modelu směr asociace označuje podřízené atributy, které je možné aktualizovat nebo odstranit společně se změnou (změnami) instance nadřazené entity (souvisí to s pojmy kaskádové události nad daty relačních databází).

Mapování dědičnosti/zobecnění

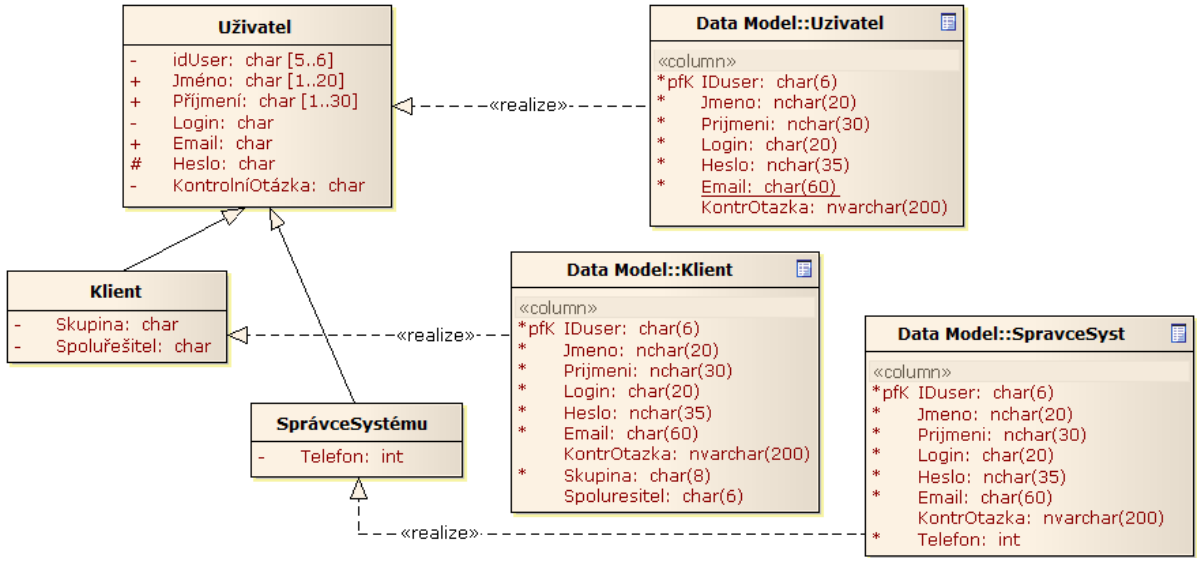
Jedním z problematických částí mapování objektového modelu do datového modelu je vztah tříd typu dědičnost. Existují tři základní postupy mapování:

Mapování celého vztahu zobecnění do jednoho datového objektu – vzniká jedna tabulka, která zahrnuje atributy všech tříd vztahu dědičnosti/zobecnění. Rozlišení jednotlivých instancí podřízených tříd je vyřešeno vytvořením nového atributu ve výsledném datovém modelu. Nevýhodou mapování do jednoho datového objektu je nevyužití všech atributů pro rozsáhlou hierarchii vztahu dědičnosti/zobecnění, kdy atributy potomků budou obsahovat nulové hodnoty.



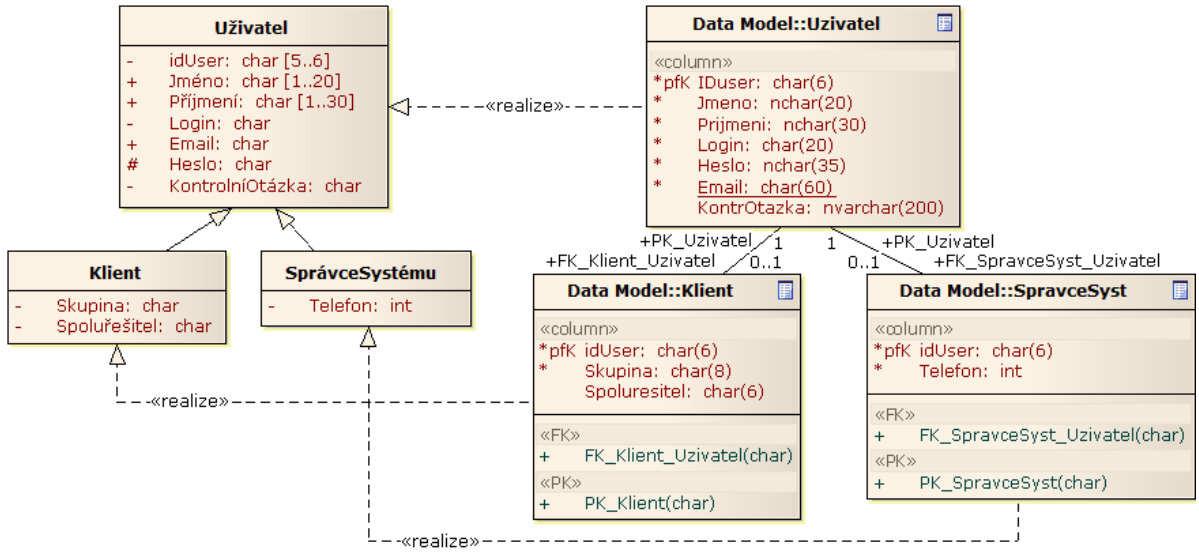
Obrázek 4.11 – Mapování celého vztahu zobecnění do jednoho datového objektu

Mapování celého vztahu zobecnění do odpovídajících datových objektů – vzniká stejný počet datových objektů jako tříd. Třídu rodiče mapujeme do entity rodiče se stejným počtem atributů. Třídy potomků mapujeme do entity potomků, v nichž kromě atributů z tříd potomků doplníme při procesu mapování také všechny atributy ze třídy rodiče. Výhodou je, že přistupujeme vždy k jedné entitě. Nevýhodou je složitost údržby datového schématu, ale také především vazba jiné entity k entitě rodiče a integrity dat vůči entitě rodiče nebo entitě potomka. V případě abstraktního rodiče nevzniká entita rodiče, ale pouze entity potomků.



Obrázek 4.12 – Mapování celého vztahu zobecnění do odpovídajících datových objektů

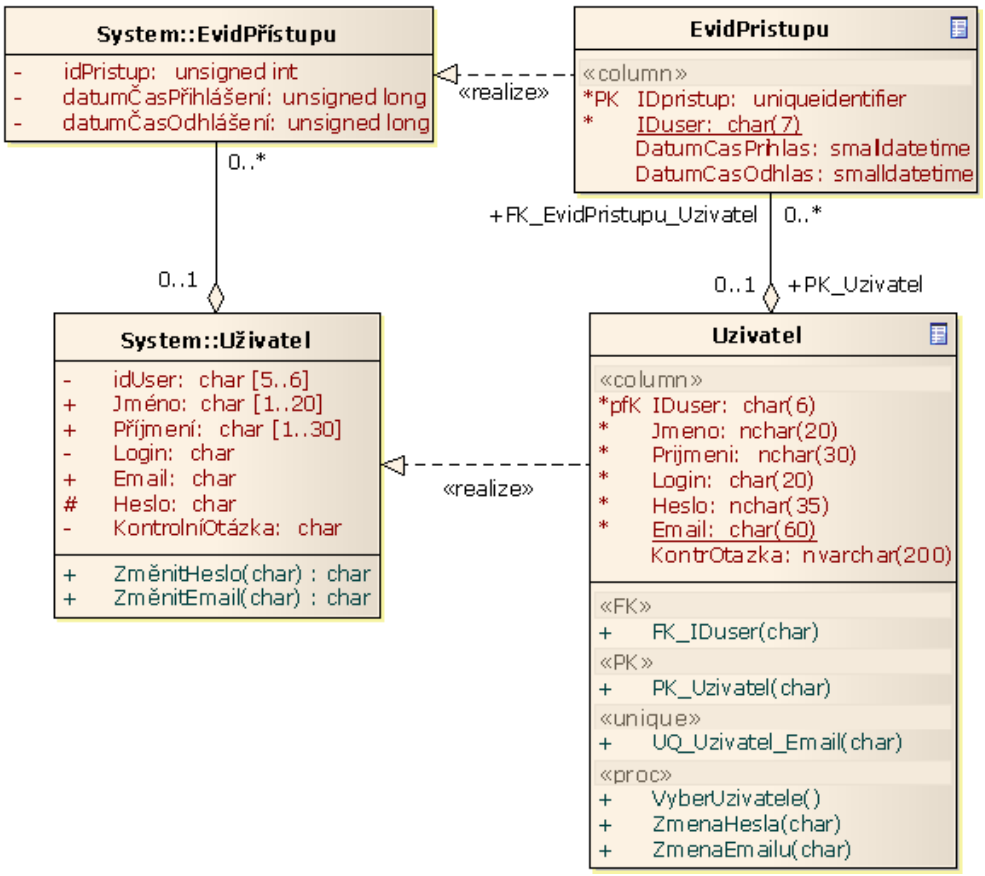
Mapování částečného vztahu zobecnění do odpovídajících datových objektů – vzniká stejný počet datových objektů jako tříd. Nově vznikající entity potomků obsahují atributy odpovídající atributům tříd potomků (specifické atributy ve vztahu dědičnosti náležící pouze danému potomku) a navíc budou entity potomků zahrnovat také primární klíč entity rodiče, což bude ve vztahu 1:1 (entita rodiče:entita potomka) představovat cizí klíč. Nevýhodou této varianty mapování je zajištění integrity dat mezi entitou potomka a entitou rodiče při vzniku (zrušení) instancí.



Obrázek 4.13 – Mapování částečného vztahu zobecnění do odpovídajících datových objektů

Mapování agregací a kompozicí

Agregace a kompozice vyjadřují vztah dvou objektů s jejich silnější vazbou na nadřazený objekt, kdy při mapování těchto vazeb nedochází ke změně násobnosti vztahu, pouze k omezení činnosti nad manipulacemi s daty. Takže asociace jsou mapovány stejným způsobem jako výše uvedené principy mapování vztahů 1:N (1:1) společně se vznikem primárních a cizích klíčů (obrázek 4.14).

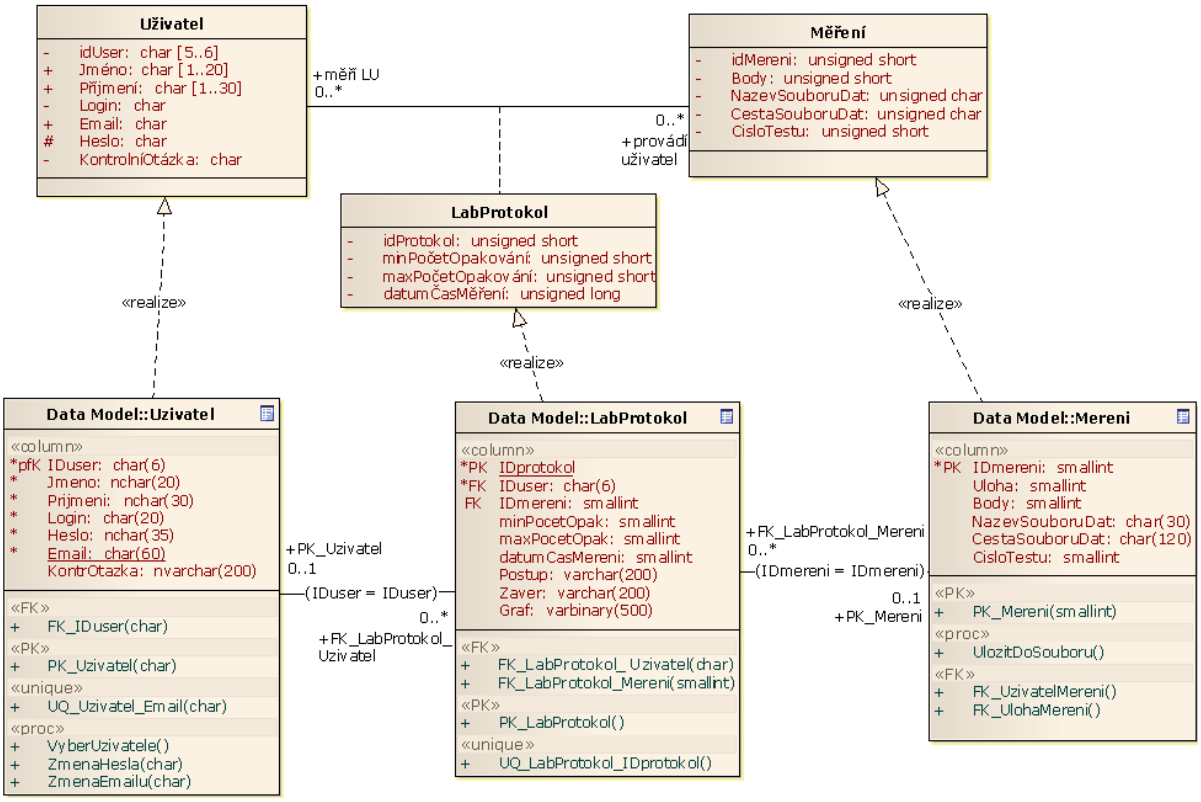


Obrázek 4.14 – Mapování kompozice

Agregace a kompozice se konkrétně odlišuje od běžných asociací tím, že se definují operace nad entitami, které charakterizují vztah agregace/kompozice. V případě kompozice platí, že při odstranění instance nadřazené entity se odstraní také všechny instance, které jsou ve vztahu k nadřazené entitě. Operace, které zajišťují odstranění instance nadřazené entity, se mapují také do entity podřízené, tedy vznikají operace stejné činnosti na obou úrovních vztahu. U vztahu agregace není vždy vztah instancí tak silný, není nutné odstranit také instance podřízené entity. Pak už vznik stejných operací na úrovni nadřazené a podřízené entity jsou individuálně řešeny.

Mapování asociačních tříd

Asociační třídy jsou specifickými objekty v diagramu tříd, které jsou vázány na asociaci. Ve vztahu asociace vznikají hodnoty a metody, které patří do asociační třídy. V relačních modelech neexistuje vazba entity s asociací, ale pouze mezi dvěma entitami. Zpravidla je mapování asociačních tříd uskutečněno vznikem nové entity a asociace mezi dvěma třídami je rozdělena na asociaci mezi nově vzniklou entitou mapovanou z asociační třídy a entitami, které jsou spojeny původní asociací vázanou na asociační třídu (obrázek 4.15).



Obrázek 4.15 – Mapování asociační třídy

Původní asociace mezi třídami byla označena M:N. Při vzniku entity mapující asociační třídu se mapují dvě nové asociační vazby 1:N a entita zahrnuje nové atributy pro mapování cizích klíčů.

Řešený příklad

Datový model může vzniknout *mapováním diagramu tříd do entit* (tabulek) nebo může být proveden *návrh datového modelu přímo z požadavků modelovaného systému*. Modelovaný systém **Evidence měření** má v nefunkčních požadavcích definováno implementační prostředí (SQL Server). Proto není nutná tvorba diagramu tříd, pokud je zřejmé, že implementace bude realizována v relačním databázovém prostředí. Z důvodu zahrnutí popisu a ukázky diagramu tříd v této publikaci byla pro tento modelovaný systém zvolena analýza nejprve obecně, objektovým modelem, a poté mapována do datového modelu.

Mapování jednotlivých tříd do entit, jejich atributů a operací bylo postupně provedeno. Příklady konkrétních situací mapování pro modelovaný systém **Evidence měření** jsou dokumentovány v předchozích obrázcích (obrázek 4.6 až obrázek 4.15).

Výsledkem objektového modelování je konceptuální a poté datový model. V části implementace pak postupně vzniká fyzický model konkrétních objektů tabulek v daném softwarovém prostředí. Postupně i ve fázi implementace dochází k doplňování a zpřesňování datového modelu, které vyplývá z realizace všech operací entit, jejich vazeb a konfrontací nad objemem i obsahem realizovaných požadavků.

Shrnutí pojmů

Objektové modelování je důležitou fází návrhu modelovaného systému, při němž vznikají postupně různé formy objektových modelů: konceptuální, datový nebo fyzický model. V rámci objektového modelování jste byli seznámeni s pojmy třída, atribut, operace, instance, asociační třída, kompozice, agregace atd. Pojmy neodmyslitelně patří k objektovému modelování a jsou základními prostředky vyjadřování databázových architektů, programátorů. Cesta od požadavků k objektovému modelu systému je poměrně náročnou činností, která kromě zhmotnění požadavků do konkrétní podoby konceptuálního modelu systému zahrnuje také využití profesních zkušeností z implementačního prostředí pro správnou volbu objektů a vazeb mezi nimi tak, aby modelovaný systém splňoval předpoklady modularity, rozšiřitelnosti a případné integrace na další systémy.

Kapitola je tedy věnována vzniku diagramu tříd, mapování diagramu tříd do datového modelu. Poskytuje přehled možných nástrojů v objektovém modelování, ale samotný postup návrhu modelovaného systému je vždy individuální a velmi klíčový pro splnění představy o funkčnosti modelovaného systému nejen z pohledu zadavatele systému, ale také pro další účastníky týmu procesu analýzy a implementace.

Otázky

1. Který z modelů: konceptuální, datový nebo fyzický je nezávislý na implementačním prostředí?
2. Vysvětlete pojem asociační třída? V jakém diagramu (diagramu tříd, datovém modelu) jej můžete najít?
3. Popište způsoby mapování vztahu dědičnosti/zobecnění do datového modelu?
4. Popište způsob, jakým se vyznačují iterace (cykly) v rámci sekvenčního diagramu, diagramu komunikací a diagramu aktivit.
5. Vysvětlete pojem „pre-condition“, „post-condition“. V jakých diagramech je analytik aplikuje?
6. Může být spojen účastník s entitní třídou přímou vazbou v sekvenčním diagramu?

CD-ROM

Analýza informačního systému je realizována v produktu Enterprise Architect, verze 8 firmy Sparx Systems Pty Ltd. Animační ukázka zahrnuje důležité části a postřehy při **realizaci diagramu tříd**. Animace je zaznamenána programem Adobe Captivate 4.0.

5 IMPLEMENTACE

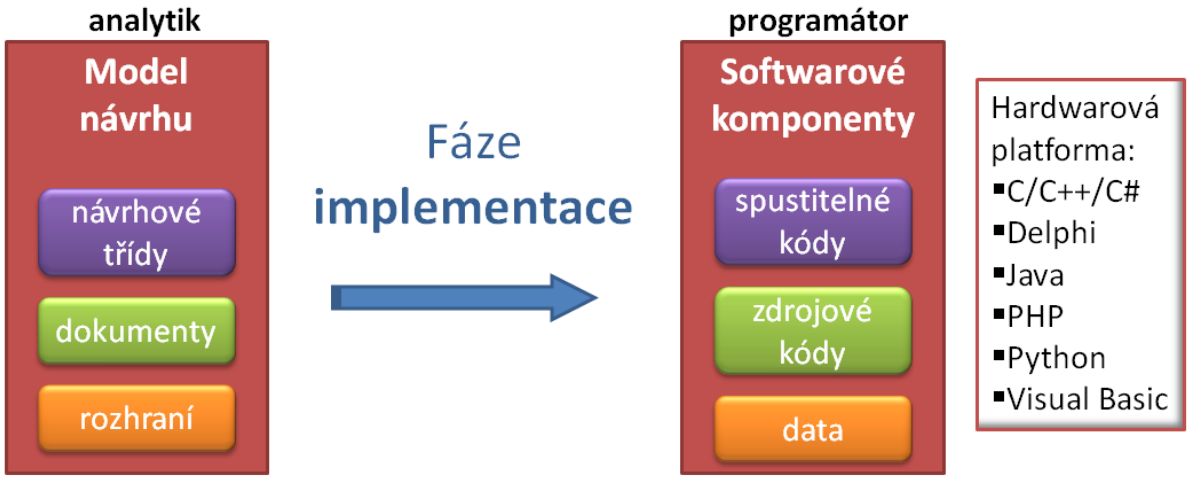
Čas ke studiu: 1 hodinu

Cíl: Po prostudování tohoto odstavce budete umět

- Definovat pojem implementační fáze.
- Vysvětlit, jak se změní diagram tříd ve fázi implementace a jaké jsou jeho výstupy.
- Modelovat diagram komponent pro konkrétní analyzovaný systém.
- Navrhovat diagram nasazení pro implementaci modelovaného systému do reálného prostředí.

Výklad

Implementační fáze procesu modelování je určena k tomu, abychom modelům (diagramům) vytvořeným ve fázi analýzy a návrhu přiřadili konkrétní podobu ve formě fyzicky existujících softwarových komponent pracujících na požadované hardwarové platformě. Modely diagramů tříd se ve fázi implementace stávají návrhovými modely tříd v daném softwarovém prostředí (obrázek 5.1).

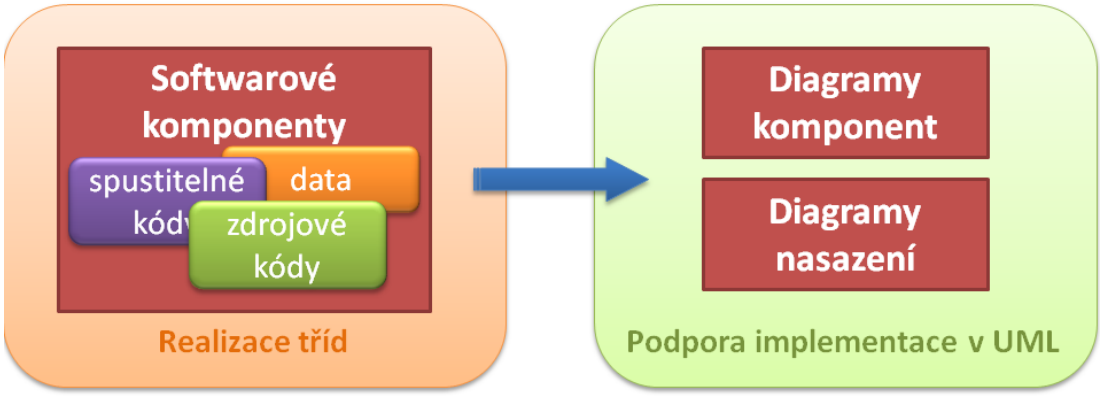


Obrázek 5.1 – Implementační fáze procesu modelování

Souhrn vytvořených komponent se distribuuje na jednotlivá zařízení, na nichž budou spuštěna. Pro zajištění bezchybného provozu programových kódů je nutné zajistit také správné rozhraní a doplnit nezbytné programové kódy.

Realizací tříd ve smyslu sestavení zdrojových kódů se zabývá programátor, zatímco analytik se zabývá tvorbou dokumentace softwarových komponent za podpory UML, které

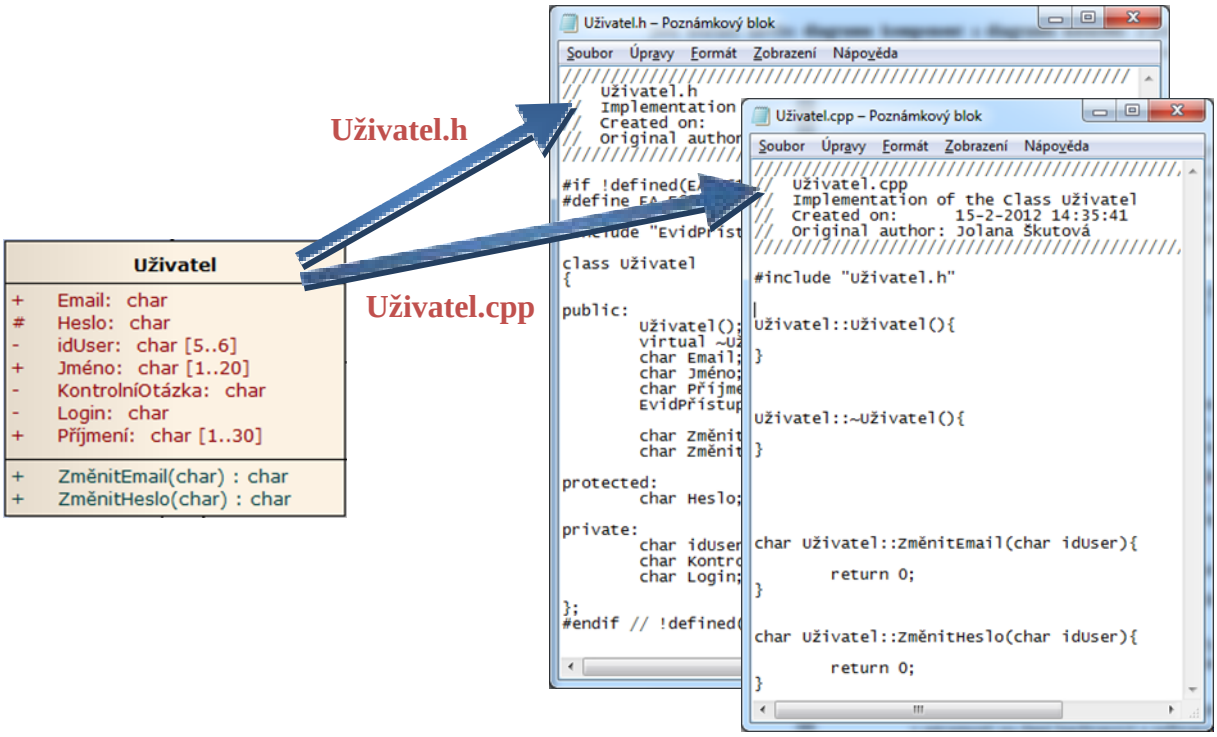
jsou součástí návrhu **diagramu komponent** a **diagramu nasazení**. Z pohledu analýzy modelovaného systému tvoří diagramy komponent a diagramy nasazení *celkový koncept fyzické implementace analýzy a návrhu modelovaného systému.*



Obrázek 5.2 – Podpora implementační fáze v procesu modelování

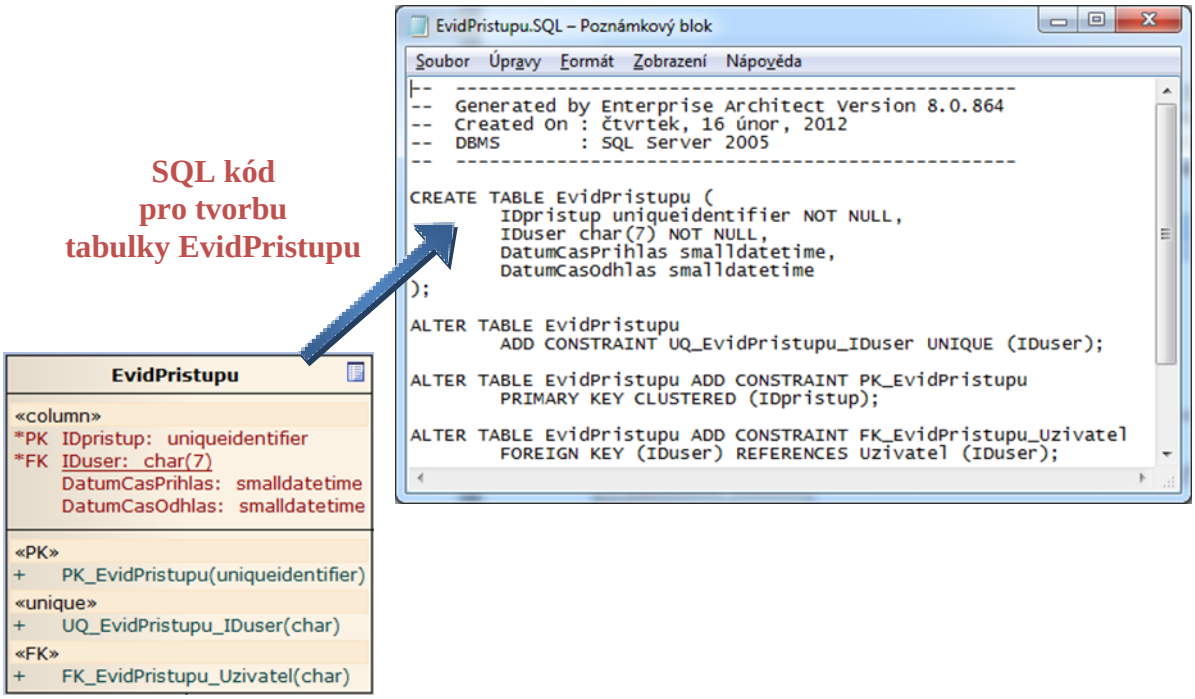
5.1 Realizace diagramů tříd

V předchozí kapitole byla definována analýza diagramu tříd, datového modelu. Implementace těchto částí analytického modelu závisí na zvoleném implementačním prostředí.



Obrázek 5.3 – Realizace diagramu tříd generováním třídy (Uživatel) do vybraného programového prostředí (C++)

V případě objektových programových prostředků (C++, C#, Java, .NET), je možné sestavit programové kódy, které implementují jednotlivé třídy, atributy a operace. V závislosti na použitém typu analytického programu je možné také tuto část, zejména třídy a jejich atributy generovat přímo z tohoto programu a rychlým způsobem tak získat základ pro jejich programovou implementaci (obrázek 5.3).



Obrázek 5.4 – Realizace datového modelu generováním spustitelných SQL příkazů, které v cílovém prostředí generují tabulky, sloupce a další prvky

V případě implementačního prostředí v podobě databází je výchozím modelem pro implementaci datový model, který rovněž dle zvoleného analytického programu lze generovat do vybraného databázového prostředí a v něm pak dále pokračovat v implementaci uložených procedur, spouští a dalších objektů relačních databází (obrázek 5.4).

Realizací diagramů tříd nebo datových modelů získá programátor zdrojové kódy, které odpovídají předchozí provedené analýze. Tím je zajištěna kompatibilita mezi analytickým návrhem a fyzickým modelem systému.

5.2 Diagram komponent

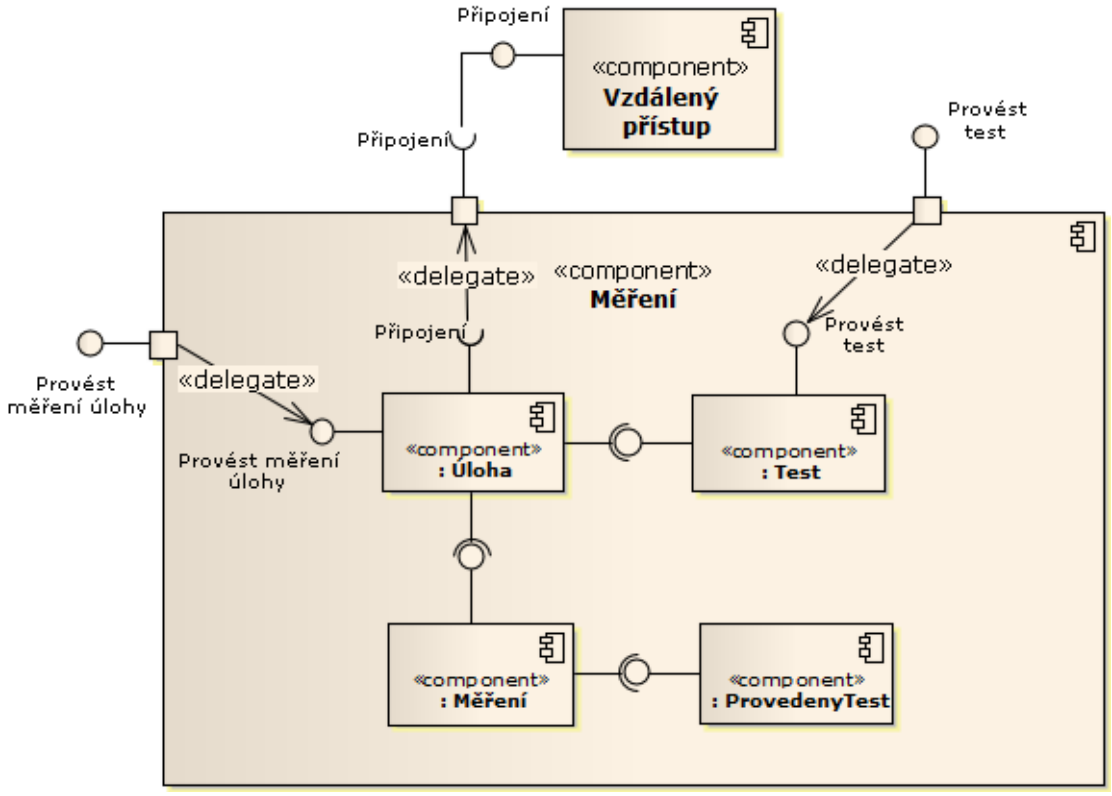
Implementace modelovaného systému je dána souborem softwarových komponent, které představují nejen návrhové modely tříd, ale také další nezbytné softwarové komponenty, kterými mohou být tabulky, dokumenty, dynamické knihovny nebo spustitelné soubory.

Koncept CBD (Component Based Development) je určen k tomu, aby využíval již vytvořených komponent k tvorbě nových modelovaných systémů a jejich kompletaci s existujících komponent, případně jejich vylepšením. S rostoucí knihovnou komponent pak klesá náročnost tvorby dalších systémů.

Komponenty v jazyku UML je možné rozdělit do dvou skupin:

- Logické komponenty – představují např. komponenty procesů a business komponenty.
- Fyzické komponenty – jsou fyzicky existující programové kódy, hlavičkové soubory např. .NET komponenty, COM+ komponenty, C++ komponenty, Java komponenty.

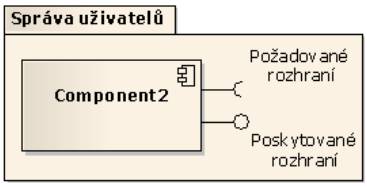
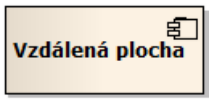
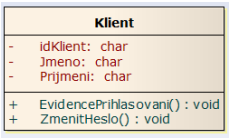
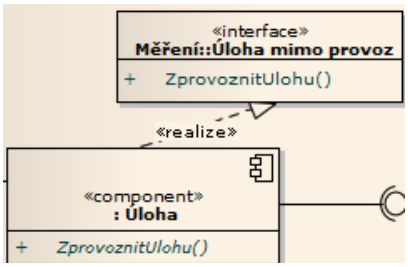
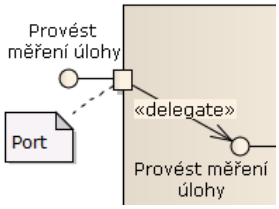
Předpokládá se, že diagramy komponent jsou určeny pro konkrétní technologie v návaznosti na dané hardwarové a softwarové prostředí.

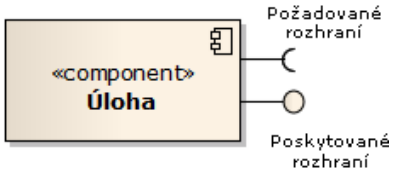
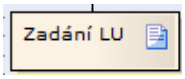
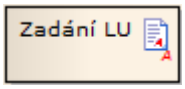


Obrázek 5.5 – Diagram komponent (pro část modelovaného systému)

Grafická implementace diagramu komponent zahrnuje různé prvky a typy relací, které jsou podrobně popsány a symbolicky zobrazeny v tab. 5.1 a tab. 5.2.

Tab. 5.1 – Grafická syntaxe a popis prvků diagramu komponent

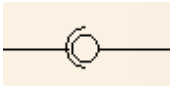
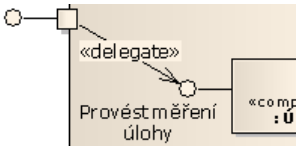
Prvky diagramu komponent	Popis
Balíček (Package) 	Logické rozvrstvení dílčích částí modelovaného systému. Balíček seskupuje dílčí elementy daného diagramu. Balíček se znázorňuje grafickou značkou, která je podobná známé ikoně pro složku. V horní části se uvádí název balíčku, který označuje typ prvků, které jsou zahrnuty do balíčku.
Komponenta (Component) 	Fyzická nahraditelná část systému, která zahrnuje implementaci a poskytuje realizaci množiny specifikovaných rozhraní. Komponenta je znázorněna obdélníkem s klíčovým slovem <<component>> a v pravém rohu uvnitř je zobrazena ikona komponenty.
Strukturovaná komponenta (Packaging Component) 	Strukturovaná komponenta může dědit a importovat své členy, např. komponenty, artefakty, třídy a vazby mezi nimi. Je zobrazena stejným způsobem jako komponenta (mírně odlišná ikona v pravém rohu), ale uvnitř této strukturované komponenty se kromě klíčového slova a jejího názvu objeví další prvky určené pro návrh diagramů komponent.
Třída (Class) 	Reprezentuje šablonu, vzor či předpis pro vznik budoucích objektů a popisuje interní strukturu těchto objektů pomocí atributů a metod dané třídy. V diagramu komponent se třída může využít jako dílčí část strukturované komponenty.
Rozhraní (Interface) 	Rozhraní je datový typ, který má jen veřejné abstraktní operace, tj. operace bez implementace. Rozhraní je specifickým typem komponenty (třídy), které je vyznačeno stereotypem <<interface>>. Obsahuje specifické atributy nebo operace.
Port (Port) 	Představuje externě viditelnou část mezi komponentou a jejím okolím. Port je nezbytnou součástí zobrazení požadovaného nebo poskytovaného rozhraní komponenty nebo jiného objektu. Je možné porty označovat názvy Port1, Port 2 nebo konkrétními názvy. Obvykle postačuje název zobrazující typ a účel požadovaného nebo poskytovaného rozhraní, které navazuje na daný port. Port může z okolí sdružovat i více než jedno rozhraní a také různých typů (požadované/poskytované).

Externí rozhraní (Expose Interface) 	Přenos informací mezi komponentou a okolím je realizován prostřednictvím dvou typů rozhraní: požadované nebo poskytované. Zpravidla se u značky daného typu rozhraní uvádí názvem jeho popis, např. ZadaniObjednavky, Ucet, ObjednanaPolozka apod. z důvodu srozumitelnosti diagramu komponent.
Artefakt (Artifact) 	Specifikuje fyzickou část informací, zdrojové nebo binární soubory, tabulky (v databázi), apod. Artefakty zpravidla navazují na komponenty vazbou označenou stereotypem <<manifest>>.
Dokument (Document Artifact) 	Dokument je specifickým artefaktem označeným stereotypem <<document>>. Prvek je graficky odlišen od prvku Artefakt symbolem *A* v pravém horním rohu.

Prvky diagramu komponent jsou vázány na port, který je nezbytnou součástí diagramu. Porty s daným názvem vyznačují činnost definovanou diagramy případů užití. Vzhledem k definovaným balíčkům se některé komponenty využívají pro různé činnosti, proto je vhodné definovat strukturovanou komponentu, která zahrnuje dílčí komponenty a porty. Strukturovaná komponenta pak může zastupovat realizovanou část komponent pro dílčí balíček.

Specifickými relacemi v diagramu komponent je montážní konektor, který je nezbytný pro definici vztahů mezi nadřazenou a podřízenou komponentou a také vazbu delegace směřující z vnějšího portu na vnitřní port označenou stereotypem <<delegate>> a konkrétně specifikovanou názvy portů, které definují požadované činnosti na komponentu.

Tab. 5.2 – Grafická syntaxe a popis relací diagramu komponent

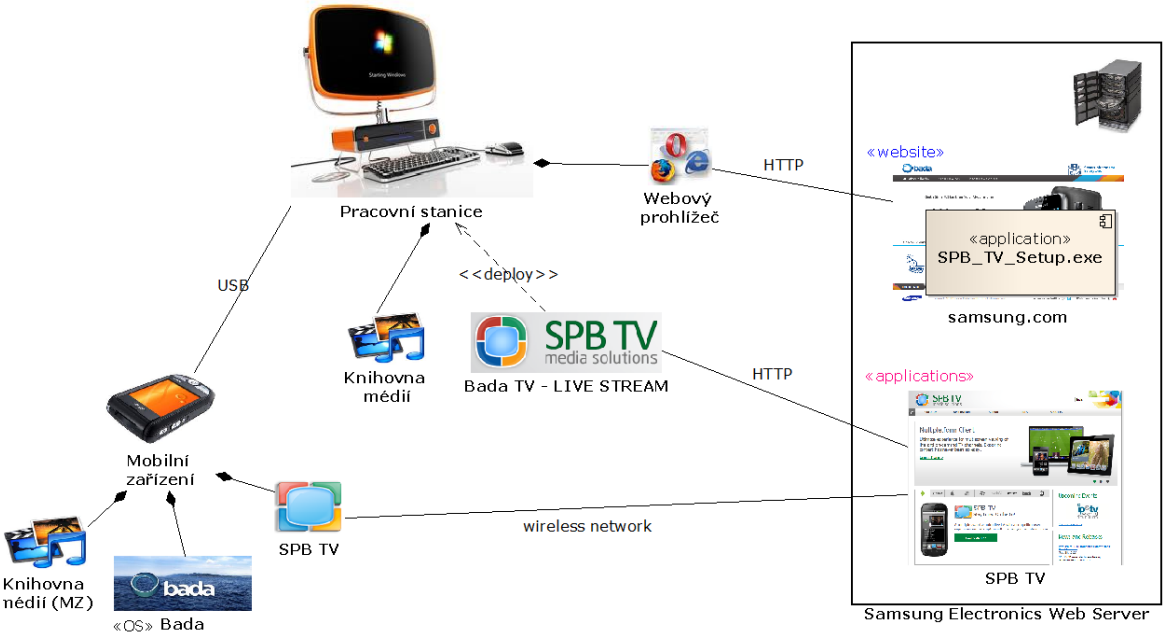
Relace mezi prvky diagramu komponent	**Popis**
Montážní konektor (Assembly) 	Montážní konektor je specifická vazba mezi dvěma komponentami, která definuje vztah, kdy jedna komponenta poskytuje „služby“ druhé komponentě. Prvkem znázorňujícím rozhraní je tzv. kulový kloub propojující dvě komponenty.
Delegace (Delegate) 	Fyzická nahraditelná část systému, která zahrnuje implementaci a poskytuje realizaci množiny specifikovaných rozhraní. Komponenta je znázorněna obdélníkem s klíčovým slovem <<component>> a v pravém rohu uvnitř je zobrazena ikona komponenty.
Asociace (Asociate)	Asociace definuje vztah mezi dvěma třídami. Graficky je zobrazena čarou rovnou nebo lomenou bez označení šipkou. Asociaci je možné přiřadit

	název dle jejího významu vztahu těchto tříd.
Realizace (Realize) «realize»	Vztah realizace existuje mezi dvěma prvky modelu, kdy jeden z nich musí implementovat nebo realizovat chování druhého prvku. Vazba je např. mezi komponentou a rozhraním.
Generalizace (Generalize)	Vztah generalizace definuje pojem dědičnost. Většinou je vázán k objektu třída nebo instance, kdy atributy a metody nadřazeného objektu jsou také atributy a metodami podřízeného objektu.

5.3 Diagram nasazení

V průběhu fáze implementace modelovaného systému se obvykle navrhují diagramy nasazení. Jsou určeny pro zobrazení fyzického uspořádání uzlů v distribuovaném systému a jejich vnitřního uspořádání v podobě artefaktů, komponent nebo další prvků. Uzly jsou zastoupeny hardwarovým zařízením, tj. počítače, sensory a tiskárny nebo také další zařízení, která podporují spustitelné prostředí systému. Jednotlivé konkrétní vazby mezi uzly nebo komunikační cesty jsou modelovány vhodnými typy relací. Kromě popisu hardwarové části informačního systému jsou pro dané uzly definovány komponenty a jejich konkretizace aplikovaného software.

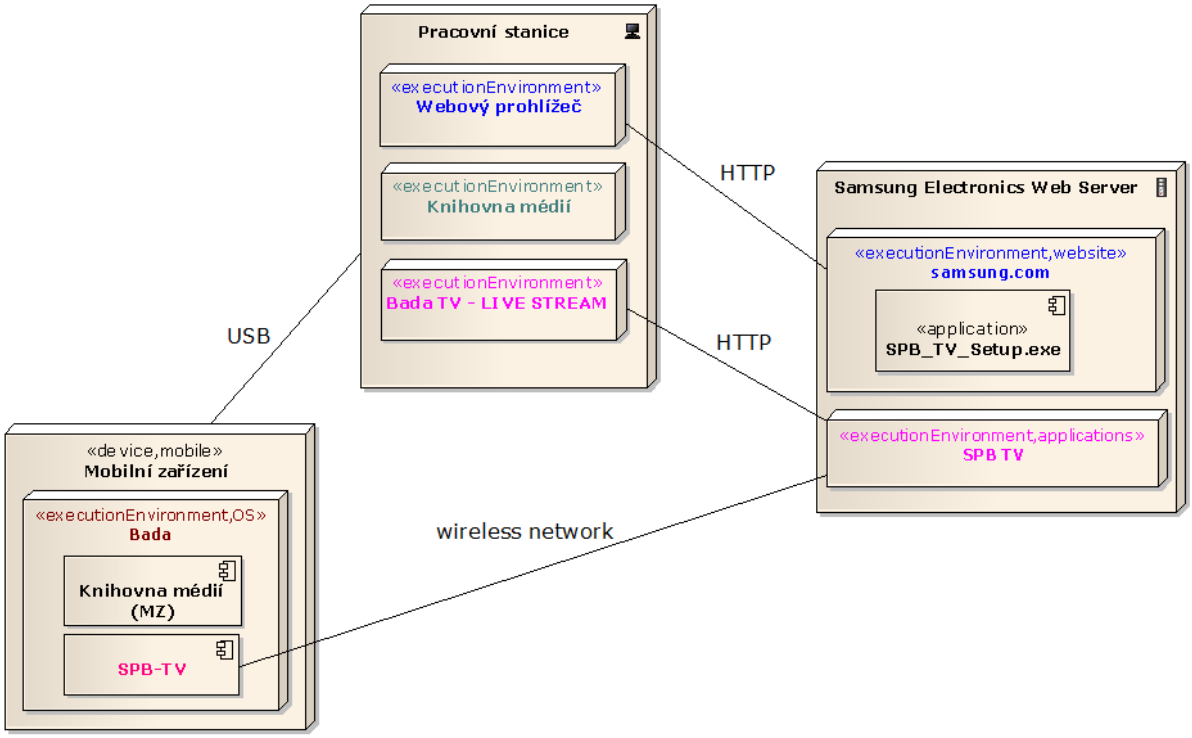
Pokud jsou známy podrobné informace o hardwaru v místě nasazení, pak je možné vytvořit diagram konkrétního nasazení, v němž se objeví konkrétní počítače s jejich konkrétním nastavením a softwarovou podporou.



Obrázek 5.6 – Diagram nasazení – aplikace správy knihovny médií na mobilním zařízení i pracovní stanici (využití obrázků pro prvky diagramu)

Diagram nasazení může být graficky zobrazen dle standardních prvků a jejich vzhledu (tab. 5.3), ale také může být lépe vizualizován nahrazením obrázků konkrétních prvků (obrázek 5.6). Nevýhodou diagramu nasazení, kde využívá uživatel obrázků konkrétních

počítačů nebo *zařízení*, je to, že nelze zobrazit *systémové prostředí* a *komponenty* jako vnořené prvky, ale je nutné je umístit vedle prvku uzel nebo zařízení s vazbou <<deploy>>. Grafické zobrazování diagramů nasazení je z hlediska vizualizace pro uživatele lépe čitelné, ale obsáhlejší pro zobrazování prvků *systémové prostředí* nebo *komponenta* vně počítače nebo zařízení. Častěji je možné prohlížet diagramy nasazení v tradičním pojetí prvků (obrázek 5.7).



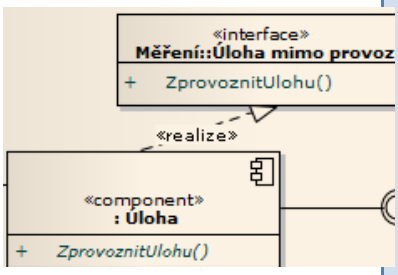
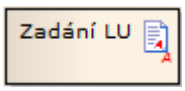
Obrázek 5.7 – Diagram nasazení – aplikace správy knihovny médií na mobilním zařízení i pracovní stanici (využití standardního vzhledu prvků diagramu)

Diagramy nasazení (Deployment Diagram) jsou výkonným nástrojem pro vizualizaci, specifikaci a dokumentaci následujících typů systémů:

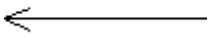
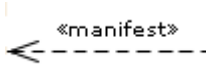
- Vestavěné systémy – využívající hardware, který je řízen externím podnětem, např. systém řízený změnou polohy fyzického zařízení.
- Systémy typu klient/server – z důvodu rozlišení uživatelského rozhraní serveru a klienta, jejich komponent a definice rozhraní.
- Distribuované systémy – za účelem definice všech zúčastněných systémů pracujících na stejných či různých platformách a pro konkretizaci vazeb mezi těmito systémy.

Pro návrh a tvorbu diagramů nasazení je možné využít prvků aplikovaných také v diagramu komponent, ale jsou zde také nové prvky. Celkový souhrn možných prvků aplikovaných v diagramech nasazení je přehledně uveden v tab. 5.3. Kromě již známých relačních vazeb, tj. asociace, generalizace, realizace, jsou v diagramu nasazení nové typy závislostí, které jsou uvedeny v tab. 5.4.

Tab. 5.3 – Grafická syntaxe a popis prvků diagramu nasazení

Prvek diagramu nasazení	Popis
Uzel (Node) 	Hardwarové zařízení, které je vizualizačně zobrazeno jako hranol v 3D prostoru. Uzly jsou rozděleny dle činnosti do dvou skupin: procesory (umožňují spuštění softwarových komponent) a zařízení (nejsou schopny samy spustit softwarovou komponentu Jedná se o fyzický objekt režimu run-time, který představuje prostředek zpracování. Uzly jsou obvykle výpočetní zařízení, mohou však představovat také lidské prostředky nebo prostředky mechanického zpracování.
Zařízení (Device)	Vyjadřuje podskupinu prvku „uzel“ a využívá se v diagramu pro grafickou interpretaci zařízení, které neumožňuje automatické spuštění softwarové komponenty, která je jeho součástí.
Systémové prostředí (Execution Environment) 	Pokud je požadována definice systémového prostředí, které je součástí uzlu nebo zařízení a je zároveň nadřazeným prvkem komponenty, pak se aplikuje prvek Systémové prostředí. Je vhodné uvést nejnižší typ a verzi systémového prostředí pro zajištění běhu informačního systému, např. Microsoft SQL Server 2005, Windows Server, IIS apod.
Komponenta (Component) 	Fyzická nahraditelná část systému, která zahrnuje implementaci a poskytuje realizaci množiny specifikovaných rozhraní. Komponenta je znázorněna obdélníkem s klíčovým slovem <<component>> a v pravém rohu uvnitř je zobrazena ikona komponenty.
Rozhraní (Interface) 	Rozhraní je datový typ, který má jen veřejné abstraktní operace, tj. operace bez implementace. Rozhraní je specifickým typem komponenty (třídy), které je vyznačeno stereotypem <<interface>>. Obsahuje specifické atributy nebo operace.
Artefakt (Artifact) 	Specifikuje fyzickou část informací, zdrojové nebo binární soubory, tabulky (v databázi), apod. Artefakty zpravidla navazují na komponenty vazbou označenou stereotypem <<manifest>>.
Dokument (Document Artifact) 	Dokument je specifickým artefaktem označeným stereotypem <<document>>. Prvek je graficky odlišen od prvku Artefakt symbolem *A* v pravém horním rohu.

Tab. 5.4 – Grafická syntaxe a popis relací diagramu nasazení (relace generalizace, realizace a delegace jsou již popsány v předchozí tabulce relací pro diagram komponent)

Relace mezi prvky diagramu nasazení	Popis
Komunikační cesta (Communication Path) 	Specifikuje spojení, které umožňuje komunikaci mezi dvěma nebo více instancemi. Komunikační cesta je specifickým typem asociace mezi uzly v diagramu nasazení, které specifikují informace, jak dochází k výměně dat a informací mezi uzly.
Evidence (Manifest) 	Vztah ukazuje, které prvky, např. třídy, instance nebo komponenty jsou zaznamenány v artefaktu. Artefakt zahrnuje konkrétní implementaci jedné nebo několika existujících softwarových komponent.
Vnoření (Nesting) 	Vazba vnoření se vyznačuje tím, že definice vnořené třídy je uvedena ve vnější třídě a cílem vnořené třídy je specifikovat další vlastnosti, které vnější třída nezahrnuje.

V diagramu nasazení se zobrazují pouze ty komponenty, které se využívají v reálném běhu programu. Ostatní komponenty se objeví pouze v diagramu komponent, např. komponenty potřebné pro překlad a sestavení spustitelných komponent.

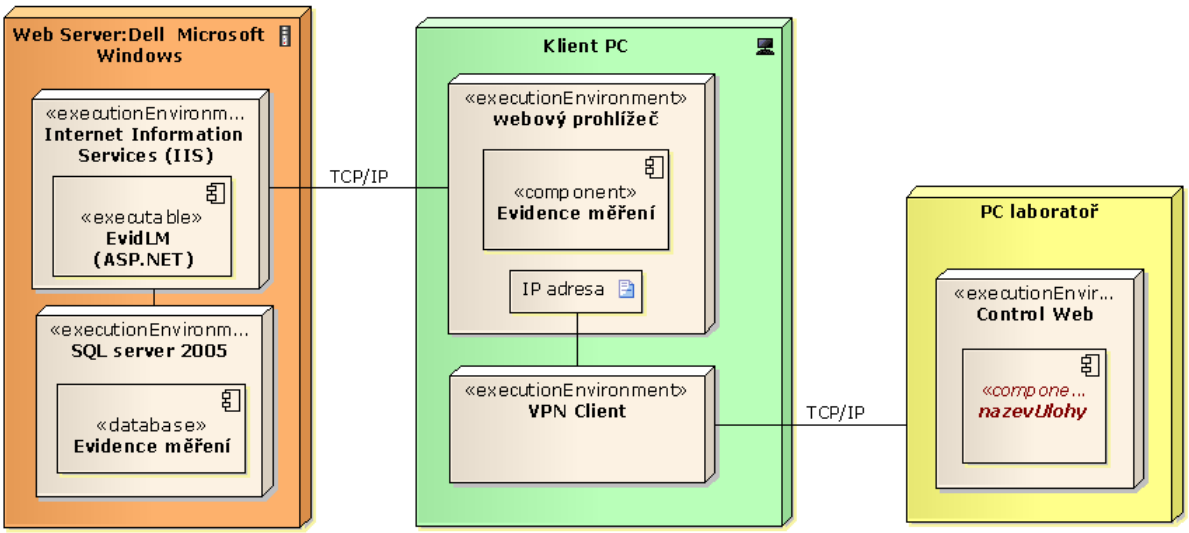
Stereotyp – zobrazuje rozdílné použití. Stereotyp je podtřídou existujícího prvku se stejnými atributy a vztahy, ale s rozdílným záměrem.

Řešený příklad

Pro informační systém Evidence měření je ve fázi implementaci nezbytné navrhnout diagram nasazení, který popisuje rozmístění komponent na daná zařízení. Pro budoucí uživatele a správce informačního systému je pak výhodou definice propojení počítačů, zařízení a dalších prvků včetně definice jejich operačních systémů, hardwarového vybavení pro bezproblémový běh systému. V počítačích je dále nezbytné instalovat potřebné softwarové vybavení, které je definováno vnořenými prvky <<executionEnvironment>>. Je vhodné společně s tímto prvkem definovat také verzi software, na níž je běh informačního systému zaručen.

Systém Evidence měření tedy definuje dva počítače (uzly), z nichž hlavním uzlem je uzel webového serveru, který zajišťuje správu dat a informací, a dále klientského počítače využívajícího systému Evidence měření z prostředí webového prohlížeče. V poznámce klientského počítače by bylo vhodné uvést typy webových prohlížečů, s nimiž je komunikace s webovým serverem ověřena. Typ a hardwarové vybavení klientského počítače je dáno požadavky programového vybavení, v daném případě je to webový prohlížeč a aplikace VPN Client. Náročnost hardwarového vybavení klientského počítače vzhledem k programovému vybavení nebude zásadně omezující a náročná ve vztahu k danému systému. Omezující by mohlo být pouze propojení počítačů přes TCP/IP rozhraní v případě spuštění měření úlohy

přes VPN Client. Je tedy možné definovat minimální zjištěné požadavky na rychlost komunikace pro úspěšné provedení měření úlohy ze vzdáleného počítače.



Obrázek 5.8 – Diagram nasazení – informační systém Evidence měření

Vytvořená aplikace zajišťující činnost systému Evidence měření je dána hlavním názvem spouštěné komponenty v daném programovém prostředí. Další komponenty, které jsou součástí navrženého systému jsou blíže specifikovány v diagramu komponent.

Shrnutí pojmů

Implementace je poslední a závěrečnou fází vývoje informačního systému, kdy uživatelům se dostává fyzické podoby navrhovaného systému. Zde se také projeví chyby v předchozí analýze, pokud nebyly objeveny v rámci průběžného testování. S implementací souvisí nejen zrození fyzických souborů a jejich instalací na cílová zařízení, ale také zajištění implementačních problémů s kompatibilitou softwarových produktů, přenosových cest. V této chvíli je nutné se ujistit, že byly splněny nefunkční požadavky, v nichž jsou přesně definovány podmínky provozu systémů.

Ve většině případů se opomíjí při závěrečném finiši návrhu a realizace informačních systémů sestavení dokumentace, která završuje celistvost vývoje informačního systému. Touto dokumentací jsou zde myšleny diagramy komponent, které zahrnují seznam a vazby mezi fyzickými soubory dat, jejich názvy, vstupními a výstupními informacemi apod. Také diagramy nasazení jsou důležité pro osoby zajišťující provoz uzlů informačního systému, přenosových cest apod. Pak při aktualizacích a změnách různých částí navrženého systému je důležité aktualizovat také diagramy nasazení a diagramy komponent.

Otázky

1. Vysvětlete pojem komponenta.
2. Jaký prvek tvoří v diagramu komponent dokument?
3. Umožňuje diagram komponent zahrnout vazbu dědičnosti?

4. Jaký prvek představuje základní část diagramu nasazení?
5. Definují prvky „Uzel“ a „Systémové prostředí“ konkrétní parametry reálného zařízení? Kde jsou uloženy tyto informace?
6. Zamyslete se nad výhodami/nevýhodami zobrazení prvku „Uzel“ v diagramu nasazení ve standardním tvaru a v grafické podobě konkrétního zařízení (počítače, serveru)?

CD-ROM

Analýza informačního systému je realizována v produktu Enterprise Architect, verze 8 firmy Sparx Systems Pty Ltd. Animační ukázka zahrnuje důležité části a postřehy při **návrhu diagramu nasazení**. Animace je zaznamenána programem Adobe Captivate 4.0.

Další zdroje

FOWLER, M. 2009. *Destilované UML : knihovna programátora*. Praha: Grada Publishing, 2009. 173 s. ISBN 978-80-247-2062-3.

KANISOVÁ, H. & MÜLLER, M. 2006. *UML srozumitelně*. Brno: Computer Press, 2006. 176 s. ISBN 80-251-1083-4.

KISZKA, B. 2003. *UML a unifikovaný proces vývoje aplikací*. Brno: Computer Press, 2003. 387 s. ISBN 80-7226-947-X.

PENDER, T. 2003. *UML bible*. Indianapolis: Wiley, 2003. 1120 s. ISBN 0-7645-2604-9.

REJNKOVÁ, P. *Příklady použití diagramů UML 2.0* [online]. 2009 vyd. [cit. 2012-05-30]. Dostupné z: http://uml.czweb.org/.

Richta, K. 2003. Unifikovaný modelovací jazyk UML. In: System Integration 2003. Praha: Česká společnost pro systémovou integraci, 2003. pp. 386-393. ISBN 80-245-0522-3.

SCHMULLER, J. 2001. *Myslíme v jazyku UML*. Praha: Grada Publishing, 2001. 359 s. ISBN 80-247-0029-8.

SPARX SYSTEMS. *Enterprise Architect* [online]. 2000-2012. [cit. 2012-05-30]. Dostupné z: http://www.sparxsystems.com/.

UML Resource Page [online]. 1997-2012. [cit. 2012-05-30]. Dostupné z: http://www.uml.org.