Systémové modelování

5

Cíle

Cílem této kapitoly je představit některé typy systémových modelů, které lze vyvinout v rámci procesů inženýrství požadavků a návrhu systému. V této kapitole:

- porozumíte tomu, jak lze reprezentovat softwarové systémy pomocí grafických modelů,
- seznámíte se s různými typy modelů a základními perspektivami systémového modelování: kontextovou, interakční, strukturní a behaviorální,
- setkáte se s některými typy diagramů jazyka UML (Unified Modeling Language) a dozvíte se, jak lze tyto diagramy uplatnit při systémovém modelování,
- získáte představu o idejích, z nichž vychází inženýrství řízené modely, kde je systém automaticky generován ze strukturních a behaviorálních modelů.

Proces systémového modelování umožňuje vyvíjet abstraktní modely systému, kdy každý model prezentuje odlišný pohled či perspektivu daného systému. Systémové modelování postupně začalo reprezentovat systémy určitým způsobem grafického zápisu, který je v současnosti téměř vždy založen na jazyku UML. Je však rovněž možné vyvinout formální (matematické) modely systému, obvykle jako podrobné specifikace systému. Grafickým modelováním pomocí jazyka UML se budeme zabývat v této kapitole a formálním modelováním v kapitole 12.

V procesu inženýrství požadavků pomáhají modely odvodit požadavky na systém, během procesu návrhu umožňují popsat systém pro techniky, kteří systém implementují, a po implementaci slouží k dokumentování struktury a fungování systému. Lze vyvinout modely stávajících systémů i nově vyvíjených systémů:

1. Modely existujícího systému se používají při inženýrství požadavků. Pomáhají vyjasnit funkce stávajícího systému a mohou sloužit jako základ pro diskusi o jeho silných a slabých stránkách. Tyto diskuse pak mohou vést k požadavkům na nový systém.
2. Modely nového systému pomáhají při inženýrství požadavků vysvětlit navržené požadavky jiným osobám zainteresovaným na systému. Technici pomocí těchto modelů rozebírají designové návrhy

a dokumentují systém pro implementaci. V procesu inženýrství řízeného modely lze z modelu systému generovat úplnou nebo částečnou implementaci systému.

Nejdůležitějším aspektem modelu systému je to, že vynechává podrobnosti. Model je abstrakce studovaného systému, nikoli jeho alternativní reprezentace. Reprezentace systému by měla v ideálním případě zachovat veškeré informace o reprezentované entitě. Abstrakce záměrně zjednodušuje a vybírá pouze nejdůležitější vlastnosti. Například ve velmi nepravděpodobném případě, že by tato kniha vycházela na pokračování v novinách, tato prezentace by byla abstrakcí hlavních témat knihy. Český překlad této knihy představuje její alternativní reprezentaci. Záměrem překladatele je zachovat veškeré informace, které obsahuje anglický originál.

Můžeme vyvinout různé modely, které budou systém reprezentovat z odlišných hledisek. Například:

1. Externí perspektiva, která modeluje kontext nebo prostředí systému.
2. Interakční perspektiva, která modeluje interakce mezi systémem a jeho prostředím nebo mezi komponentami systému.
3. Strukturní perspektiva, která modeluje uspořádání systému nebo strukturu dat, která systém zpracovává.
4. Behaviorální perspektiva, která modeluje dynamické chování systému a to, jak systém reaguje na události.

Tyto perspektivy mají hodně společných prvků s Krutchenovým pohledem 4 + 1 na systémovou architekturu (Kruchten, 1995), v němž navrhuje dokumentovat architekturu a organizaci systému z různých perspektiv. Uvedeným přístupem 4 + 1 se budeme zabývat v kapitole 6.

V této kapitole používáme diagramy definované v UML (Booch et al., 2005; Rumbaugh et al., 2004), který se etabloval jako standardní modelovací jazyk pro objektově orientované modelování. Jazyk UML obsahuje mnoho typů diagramů, a umožňuje proto vytvářet mnoho různých typů systémových modelů. Studie z roku 2007 (Erickson a Siau, 2007) však ukázala, že podle názoru většiny uživatelů jazyka UML lze základní aspekty systému reprezentovat pomocí pěti typů diagramů:

1. Diagramy aktivity, které znázorňují aktivity související s procesem nebo zpracováním dat.
2. Diagramy případů použití, které zobrazují interakce mezi systémem a jeho prostředím.
3. Sekvenční diagramy, které představují interakce mezi aktéry a systémem a mezi komponentami systému.
4. Diagramy tříd, které ukazují objektové třídy v systému a asociace těchto tříd.
5. Stavové diagramy, které popisují, jak systém reaguje na interní a externí události.

Vzhledem k tomu, že zde nemáme místo na rozbor všech typů diagramů UML, zaměříme se na těchto pět klíčových typů diagramů, které se používají v systémovém modelování.

Při vývoji systémových modelů lze často způsob použití grafických značek pružně přizpůsobovat. Pokaždé není nutné se přísně držet detailů dané notace. Úroveň podrobností a striktnost modelu závisí na tom, jak chceme s modelem pracovat. Grafické modely se běžně používají třemi způsoby:

1. Jako základ pro diskuse o existujícím nebo navrženém systému.
2. Jako způsob dokumentace stávajícího systému.
3. Jako podrobný popis systému, pomocí něhož lze generovat implementaci systému.

V prvním případě je účelem modelu stimulovat diskusi mezi softwarovými inženýry, kteří se podílejí na vývoji systému. Modely nemusí být úplné (za předpokladu, že zahrnují klíčová témata diskuse) a modelovací notaci mohou používat neformálně. Tímto způsobem se modely obvykle uplatňují v takzvaném „agilním modelování" (Ambler a Jeffries, 2002). Když modely slouží k dokumentaci, nemusí být úplné, protože účelem může být vývoj modelu pouze pro některé části systému. Tyto modely však musí být správné – měly by správným způsobem používat notaci a přesně popisovat systém.

JAZYK UML

Jazyk UML (Unified Modeling Language) je sada 13 různých typů diagramů, které umožňují modelovat softwarové systémy. Vznikl v 90. letech na základě práce na objektově orientovaném modelování, kdy došlo k integraci podobných objektově orientovaných notací. Hlavní revize (UML 2) byla dokončena roku 2004. Jazyk UML je univerzálně uznáván jako standardní přístup k vývoji modelů softwarových systémů. Objevily se také návrhy pro obecnější systémové modelování.

http://www.SoftwareEngineering-9.com/Web/UML/

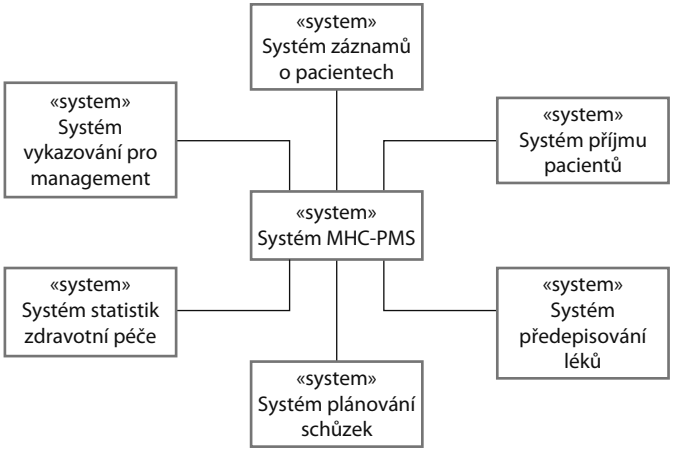
Ve třetím případě, kdy se modely používají v rámci vývojového procesu založeného na modelech, musí být systémové modely úplné i správné. Důvod spočívá v tom, že se podle těchto modelů generuje zdrojový kód systému. Je proto nutné věnovat mimořádnou pozornost tomu, aby nedošlo k záměně podobných symbolů, jako jsou šipky s čárovými a plnými hroty, které mají odlišný význam.

5.1 Kontextové modely

V rané fázi specifikace systému se musíme rozhodnout, kde budou hranice systému. Přitom musíme spolupracovat s osobami zainteresovanými na systému a určit, které funkce bude systém zahrnovat a co bude poskytovat prostředí systému. Můžeme například požadovat, že pro některé podnikové procesy bude implementována automatizovaná podpora, ale další procesy budou fungovat manuálně nebo jejich podporu zajistí jiné systémy. Měli bychom zjistit, kde se funkčnost systému překrývá se stávajícími systémy, a zvolit, zda se budou nové funkce implementovat. Tato rozhodnutí je potřeba přijmout v rané fázi celého procesu, aby se omezily náklady a zkrátil čas na analýzu systémových požadavků a návrhu.

V některých případech vede mezi systémem a jeho prostředím relativně jasná hranice. Pokud například automatizovaný systém nahrazuje stávající ruční nebo počítačový systém, prostředí nového systému zpravidla odpovídá prostředí existujícího systému. V jiných případech je k dispozici větší flexibilita a přesnou hranici mezi systémem a jeho prostředím lze určit během procesu inženýrství požadavků.

Předpokládejme například, že vyvíjíme specifikaci informačního systému o psychiatrických pacientech. Tento systém má uchovávat informace o pacientech navštěvujících psychiatrické kliniky a o léčbě, která jim byla předepsána. Při vývoji specifikace tohoto systému je potřeba se rozhodnout, zda se má systém zaměřit výhradně na shromažďování informací o konzultacích (kdy se osobní informace o pacientech budou získávat z jiných systémů), nebo zda bude rovněž shromažďovat osobní data pacientů. Pokud se informace o pacientech získávají z jiných systémů, je to výhodné z toho hlediska, že nedochází k duplikování dat. Hlavní nevýhoda však spočívá v tom, že použití jiných systémů může zpomalit přístup k informacím. Jestliže tyto systémy nejsou dostupné, nelze systém MHC-PMS použít.



Obrázek 5.1 – Kontext systému MHC-PMS

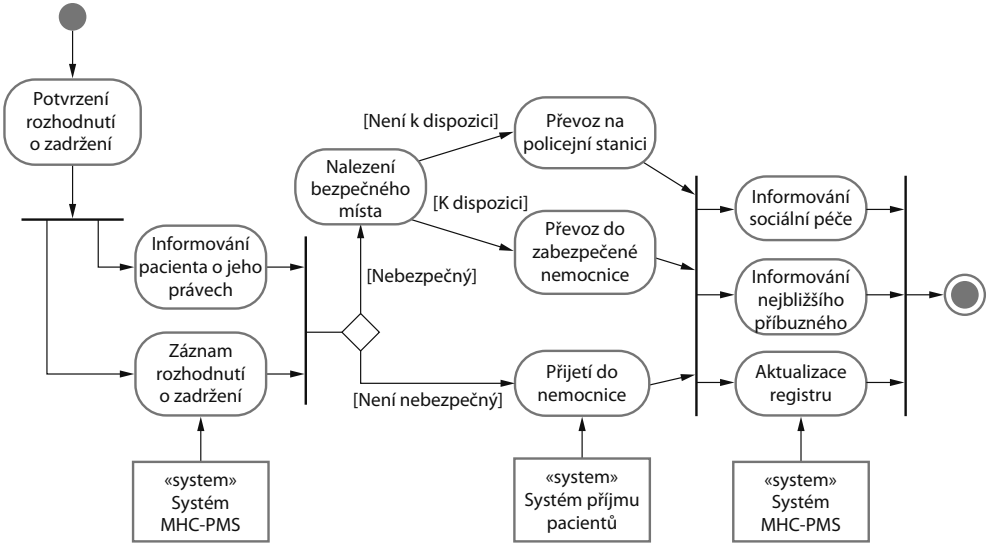
Systémovou hranici nelze definovat nezávisle na hodnotách. Vzhledem ke společenským a organizačním hlediskům může být pozice systémové hranice určena mimotechnickými faktory. Systémová hranice může být například záměrně umístěna tak, aby bylo možné analytický proces kompletně zajistit v jedné lokalitě, může být zvolena tak, aby nebylo nutné konzultovat konkrétního manažera, se kterým se obtížně spolupracuje, může být stanovena tak, aby se zvýšily náklady na systém, a bylo tedy nutné rozšířit divizi systémového vývoje, která systém navrhuje a implementuje.

Jakmile jsou přijata rozhodnutí ohledně hranic systému, je potřeba v rámci analytické aktivity definovat kontext a závislosti systému na jeho prostředí. Prvním krokem této aktivity zpravidla bývá vytvoření jednoduchého modelu architektury.

Obrázek 5.1 znázorňuje jednoduchý kontextový model, který představuje informační systém o pacientech a jiné systémy v jeho prostředí. Z obrázku 5.1 je patrné, že systém MHC-PMS je připojen na systém plánování schůzek a obecnější systém záznamů o pacientech, se kterým sdílí data. Systém je také připojen k systémům na správu vykazování a přidělování nemocničních lůžek a statistickému systému, který shromažďuje informace pro výzkum. Nakonec používá i systém předepisování léků, který pacientům generuje recepty na jejich léky.

Kontextové modely obvykle ukazují, že prostředí obsahuje několik dalších automatizovaných systémů. Neposkytují však informace o typech vztahů mezi systémy v prostředí a o specifikovaném systému. Externí systémy mohou poskytovat data pro daný systém nebo přijímat data z tohoto systému. Mohou se systémem sdílet data nebo je možné připojit je přímo či prostřednictvím sítě a také nemusí být připojeny vůbec. Někdy se nacházejí na stejném místě, případně mohou být v různých budovách. Všechny tyto vztahy mohou mít vliv na požadavky a návrh definovaného systému a je potřeba je zohlednit.

Jednoduché kontextové modely se proto používají spolu s jinými modely, jako jsou například modely obchodních procesů. Tyto modely popisují ruční a automatizované procesy, v nichž se určité softwarové systémy používají.



Obrázek 5.2 – Procesní model nedobrovolného zadržení

Obrázek 5.2 modeluje důležitý systémový proces, který ukazuje způsob, jakým se systém MHC-PMS používá. Pacienti, kteří trpí duševními problémy, mohou být někdy nebezpeční sobě i jiným. Může být proto nutné zadržet je v nemocnici proti jejich vůli, aby je bylo možné léčit. Toto zadržení podléhá přísným zákonným omezením. Rozhodnutí o zadržení pacienta je například nutné pravidelně revidovat, aby lidé nebyli bez dostatečného důvodu dlouhodobě omezováni na svobodě. Jednou z funkcí systému MHC-PMS je kontrolovat, zda jsou implementovány takové zákonné pojistky.

Na obrázku 5.2 vidíme diagram aktivity UML. Diagramy aktivity mají znázornit aktivity, ze kterých sestává systémový proces, a postup, kterým si jednotlivé aktivity vzájemně předávají řízení. Začátek procesu je symbolizován vyplněným kruhem a konec je znázorněn vyplněným kruhem uvnitř jiného kruhu. Obdélníky se zaoblenými rohy reprezentují aktivity, tzn. konkrétní dílčí procesy, které je potřeba provést. Do grafů aktivity lze umístit i objekty. Na obrázku 5.2 jsou znázorněny systémy, které se používají na podporu různých procesů. Z funkce stereotypu UML je patrné, že se jedná o samostatné systémy.

Šipky v diagramu aktivity UML představují tok činností od jedné aktivity k jiné. Plná čára znamená koordinaci aktivit. Když tok z více aktivit vede k plné čáře, pak další postup vyžaduje, aby byly dokončeny všechny tyto aktivity. Když tok od plné čáry směřuje k více aktivitám, lze tyto aktivity provádět paralelně. Na obrázku 5.2 je tedy patrné, že aktivity informování sociální péče a nejbližšího příbuzného pacienta a aktualizace registru zadržení mohou probíhat zároveň.

K šipkám je možné doplnit popisky informující o podmínce, za které daný tok probíhá. Na obrázku 5.2 se nacházejí popisky, které označují toky pro pacienty podle toho, zda jsou nebo nejsou nebezpeční pro společnost. Pacienty nebezpečné pro společnost je potřeba zadržet v zabezpečeném zařízení. Pacienty se sebevražednými sklony, kteří jsou tedy nebezpeční sobě samým, lze však zadržet na vhodném nemocničním oddělení.

5.2 Modely interakcí

Ve všech systémech dochází k nějakému druhu interakcí. Může se jednat o uživatelskou interakci, která spočívá v uživatelském vstupu a výstupu, interakci vyvíjeného systému s jinými systémy nebo interakci mezi komponentami systému. Modelování uživatelské interakce je důležité, protože pomáhá identifikovat uživatelské požadavky. Modelování interakce mezi systémy ukazuje, jaké problémy mohou nastat při komunikaci. Modelování interakce komponent pomáhá porozumět tomu, zda navržená struktura systému pravděpodobně zajistí požadovanou úroveň výkonu a spolehlivosti systému.

V této části se budeme zabývat dvěma souvisejícími přístupy k modelování interakcí:

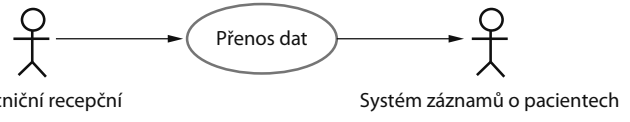
1. Modelování případů použití, které se používá zejména při modelování interakcí mezi systémem a externími aktéry (uživateli nebo jinými systémy).
2. Sekvenční diagramy, které slouží k modelování interakcí mezi systémovými komponentami, ačkoli mohou zahrnovat i externí agenty.

Modely případů použití a sekvenční diagramy předkládají interakce na různé úrovni podrobností, a lze je proto používat společně. Sekvenční diagram může dokumentovat podrobnosti interakcí, které se účastní na vysokoúrovňovém případu použití. Jazyk UML zahrnuje také komunikační diagramy, které dovolují modelovat interakce. Těmto diagramům se zde nebudeme věnovat, protože představují alternativní reprezentaci sekvenčních grafů. Některé nástroje dokážou v praxi generovat komunikační diagram ze sekvenčního diagramu.

5.2.1 Modelování případů použití

Modelování případů použití původně vyvinul Jacobson et al. (1993) v 90. letech a toto modelování se dostalo do první verze jazyka UML (Rumbaugh et al., 1999). Jak jsme diskutovali v kapitole 4, modelování případů použití se široce používá na podporu zjišťování požadavků. Případ použití lze považovat za jednoduchý scénář, který popisuje, co uživatel očekává od systému.

Každý případ použití reprezentuje diskrétní úkol, který zahrnuje externí interakci se systémem. Ve své nejjednodušší formě je případ použití znázorněn formou elipsy. Aktéři, kteří se na případu použití podílejí, jsou pak reprezentováni schematickými figurkami. Obrázek 5.3 představuje případ použití systému MHC-PMS. Jedná se o úkol odeslání dat ze systému MHC-PMS do obecnějšího systému záznamů o pacientech. Tento obecnější systém udržuje souhrnná data o pacientech a nikoli data o jednotlivých konzultacích, která jsou ukládána do systému MHC-PMS.



Obrázek 5.3 – Případ použití přenosu dat

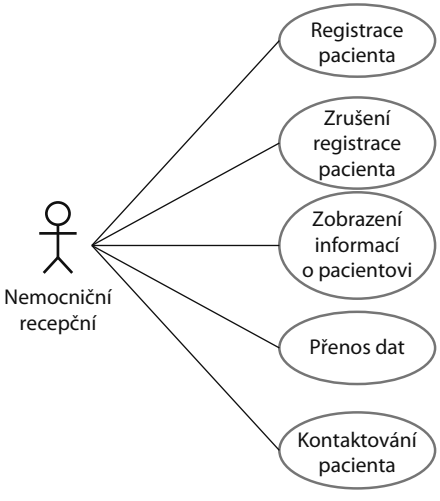
Všimněme si, že v tomto případu použití vystupují dva aktéři: operátor, který přenáší data, a systém záznamů o pacientech. Notace se schematickými panáčky měla sice původně znázorňovat interakci osob, ale nyní umožňuje reprezentovat i jiné externí systémy a hardware. Diagramy případů použití by formálně měly používat čáry bez šipek, protože šipky v jazyce UML označují směr toku zpráv. V případech použití jsou zprávy samozřejmě přenášeny oběma směry. Šipky na obrázku 5.3 však neformálně umožňují označit, že transakci inicializuje nemocniční recepční a data se přenášejí do systému záznamů o pacientech.

Diagramy případů použití poskytují poměrně jednoduchý přehled o interakci. Chceme-li tedy rozumět souvislostem, potřebujeme více podrobností. Tyto detaily mohou mít formu jednoduchého textového popisu, strukturovaného popisu v tabulce nebo sekvenčního diagramu (viz následující rozbor). Nejvhodnější formát můžeme zvolit v závislosti na případu použití a na úrovni podrobností, kterou podle našeho názoru model vyžaduje. Podle názoru autora je nejužitečnější standardní tabulková forma. Obrázek 5.4 představuje tabulkový popis případu použití „Přenos dat".

Jak jsme zmínili v kapitole 4, složené diagramy případů použití znázorňují více různých případů použití. Někdy lze do jediného složeného diagramu případů použití zahrnout všechny možné interakce v systému. Jindy to však vzhledem k počtu případů použití nemusí být možné. V takových scénářích můžeme vytvořit několik diagramů, z nichž každý představuje související případy použití. Obrázek 5.5 například znázorňuje všechny případy použití v systému MHC-PMS, na kterých se podílí aktér „Nemocniční recepční".

MHC-PMS: PŘENOS DAT	
Aktéři	Nemocniční recepční, systém záznamů o pacientech (PRS)
Popis	Recepční může přenést data ze systému MHC-PMS do databáze univerzálního systému záznamů o pacientech, který provozuje ministerstvo zdravotnictví. Přenášet lze buď aktualizované osobní údaje (adresa, telefonní číslo atd.) nebo souhrnné informace o diagnóze a léčení pacienta.
Data	Osobní údaje o pacientovi, souhrn léčby
Impuls	Uživatelský příkaz zadaný nemocničním recepčním
Odpověď	Potvrzení o aktualizaci systému PRS
Poznámky	Recepční musí mít odpovídající bezpečnostní oprávnění pro přístup k informacím o pacientovi a do systému PRS.

Obrázek 5.4 – Tabulkový popis případu použití „Přenos dat"



Obrázek 5.5 – Případy použití související s rolí „nemocniční recepční"

5.2.2 Sekvenční diagramy

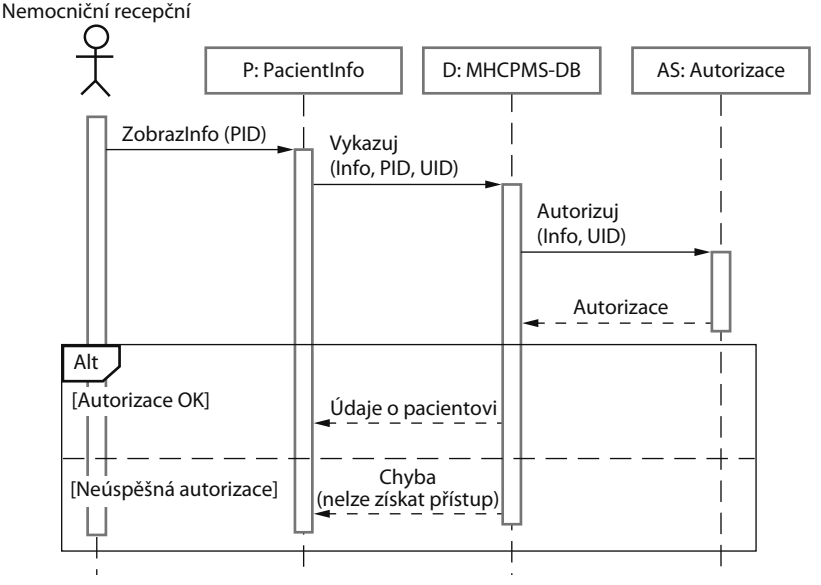
Sekvenční diagramy v jazyce UML primárně slouží k modelování interakcí mezi aktéry a objekty v systému a mezi samotnými objekty. Jazyk UML poskytuje bohatou syntaxi sekvenčních diagramů, díky níž lze modelovat mnoho různých typů interakcí. Nemáme zde místo, abychom se zabývali všemi možnostmi, takže zůstaneme u základů tohoto typu diagramu.

Jak vyplývá z názvu, sekvenční diagram znázorňuje pořadí interakcí, k nimž dochází během určitého případu použití nebo instance případu použití. Obrázek 5.6 představuje příklad sekvenčního diagramu, z něhož lze vysledovat základy příslušné notace. Tento diagram modeluje interakce, které se uplatňují v případu použití Zobrazení pacienta, kdy může nemocniční recepční zobrazit některé údaje o pacientovi.

Zúčastněné objekty a aktéři jsou uvedeni v horní části diagramu a svisle od nich vede tečkovaná čára. Interakci mezi objekty symbolizují šipky s popisky. Obdélník na tečkované čáře znamená životní čáru dotčeného objektu (tj. čas, kdy se instance objektu účastní na výpočtu). Sekvenci interakcí lze číst shora dolů. Poznámky na šipkách popisují volání objektů, jejich parametry a návratové hodnoty. V tomto příkladu je také znázorněn zápis označující alternativy. Používá se rámeček s názvem „alt" s podmínkami, které jsou uzavřeny v hranatých závorkách.

Obrázek 5.6 lze číst takto:

1. Nemocniční recepční aktivuje metodu ZobrazInfo v instanci P třídy objektů PacientInfo a poskytne identifikátor pacienta PID. P je objekt uživatelského rozhraní, který se zobrazuje jako formulář s údaji o pacientovi.
2. Instance P zavolá databázi, která vrátí požadované informace. Přitom poskytne identifikátor recepčního, který umožní bezpečnostní kontrolu (v této fázi se nezajímáme o to, odkud tento identifikátor UID pochází).



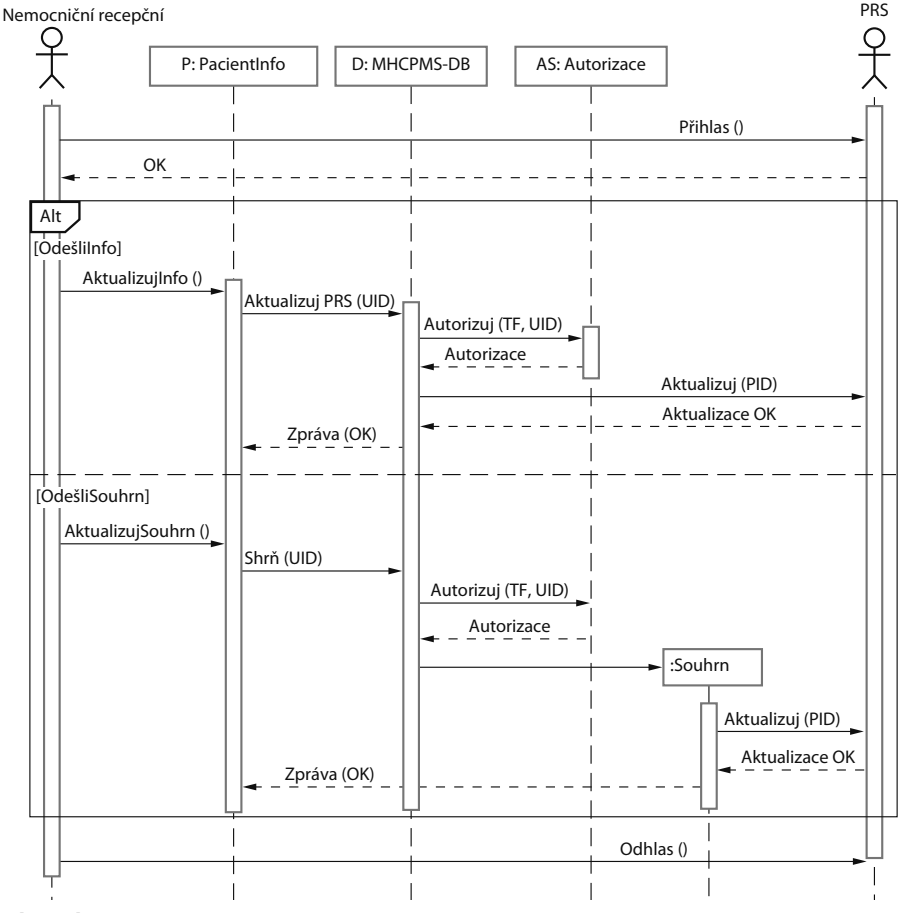
Obrázek 5.6 – Sekvenční diagram pro Zobrazení informací o pacientovi

3. Databáze pomocí autorizačního systému zkontroluje, zda je uživatel oprávněn provést danou akci.

4. Pokud ano, jsou vráceny informace o pacientovi, které vyplní formulář na obrazovce uživatele. Jestliže autorizace není úspěšná, zobrazí se chybová zpráva.

Obrázek 5.7 je dalším příkladem sekvenčního diagramu ze stejného systému, který ilustruje dvě další funkce. Jedná se o přímou komunikaci mezi aktéry v systému a vytvoření objektů jako součást sekvence operací. V tomto příkladu je vytvořen objekt typu Souhrn, který uchovává souhrnná data pro odeslání do systému PRS (systému záznamů o pacientech). Tento diagram lze číst následovně:

1. Recepční se přihlásí do systému PRS.
2. K dispozici jsou dvě možnosti. Dovolují přímý přenos aktualizovaných údajů o pacientovi do systému PRS a přenos souhrnných zdravotních dat ze systému MHC-PMS do systému PRS.
3. V obou případech se pomocí autorizačního systému kontrolují oprávnění recepčního.
4. Osobní informace lze přenášet přímo z objektu uživatelského rozhraní do systému PRS. Případně je možné vytvořit souhrnný záznam z databáze a následně přenést tento záznam.
5. Při dokončení přenosu vydá systém PRS stavovou zprávu a uživatel se odhlásí.



Obrázek 5.7: Sekvenční diagram pro přenos dat

Pokud sekvenční diagramy neslouží ke generování kódu nebo k podrobnému dokumentování, nemusí zahrnovat všechny interakce. Vyvíjíme-li na začátku vývojového procesu systémové modely, které budou podporovat inženýrství požadavků a vysokoúrovňový návrh, budou obsahovat mnoho interakcí, které závisí na volbách při implementaci. Například na obrázku 5.7 lze odložit rozhodnutí o tom, jak se bude získávat identifikátor uživatele při kontrole autorizace. Při implementaci se může uplatnit interakce s objektem Uživatel, ale v této fázi to není důležité, a proto to do sekvenčního diagramu není nutné umístit.

OBJEKTOVĚ ORIENTOVANÁ ANALÝZA POŽADAVKŮ

Při objektově orientované analýze požadavků se entity reálného světa modelují pomocí objektových tříd. Můžeme vytvářet různé typy objektových modelů, které ukazují, jak vzájemně souvisejí objektové třídy, jak lze agregací objektů vytvářet jiné objekty, jak objekty interagují s jinými objekty atd. Každý z těchto modelů předkládá jedinečné informace o specifikovaném systému.

http://www.SoftwareEngineering-9.com/Web/OORA/

5.3 Strukturní modely

Strukturní modely softwaru znázorňují organizaci systému z hlediska komponent, které systém tvoří, a jejich vztahů. Strukturní modely mohou být statické, které představují strukturu návrhu systému, nebo dynamické, které ukazují organizaci systému při jeho činnosti. Jedná se o různé pohledy – dynamická organizace systému jako sady interagujících vláken se může značně lišit od statického modelu systémových komponent.

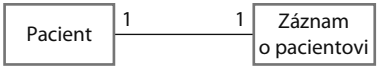
Strukturní modely systému lze vytvořit během diskuse a návrhu systémové architektury. Návrh architektury představuje zvláště důležité téma softwarového inženýrství. Při prezentaci modelů architektury lze použít diagramy komponent, balíčků a nasazení jazyka UML. Jednotlivými aspekty softwarové architektury a modelování architektury se budeme zabývat v kapitolách 6, 18 a 19. V této části se zaměříme na použití diagramů tříd při modelování statické struktury objektových tříd v softwarovém systému.

5.3.1 Diagramy tříd

Diagramy tříd se používají při vývoji objektově orientovaného modelu systému. Ukazují třídy v systému a jejich asociace. Objektovou třídu si můžeme volně představit jako obecnou definici jednoho druhu systémového objektu. Asociace (přidružení) je vazba mezi třídami, která informuje o tom, že tyto třídy nějak souvisejí. Každá třída proto může mít určité informace o svých přidružených třídách.

Při vývoji modelů během raných fází procesu softwarového inženýrství reprezentují objekty určité prvky reálného světa, jako je pacient, recept, lékař atd. Jak postupuje vývoj implementace, obvykle je nutné definovat její další objekty, které poskytují požadované funkce systému. Zde se soustředíme na modelování objektů reálného světa v rámci zpracování požadavků nebo procesů počátečních návrhů softwaru.

Diagramy tříd v jazyce UML lze vyjádřit na různé úrovni podrobností. Když vyvíjíme model, v první fázi se obvykle podíváme do reálného světa, identifikujeme klíčové objekty a reprezentujeme je formou tříd. Nejjednodušší způsob, jak třídy znázornit, je zapsat jejich název do rámečku. Můžeme také jednoduše zaznamenat existenci přidružení tak, že mezi třídami nakreslíme čáru. Obrázek 5.8 například představuje jednoduchý diagram tříd, který znázorňuje dvě třídy: Pacient a Záznam o pacientovi, mezi nimiž existuje asociace.



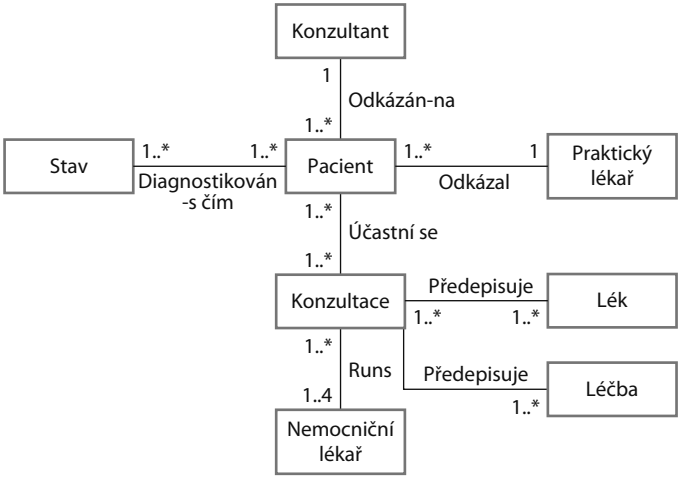
Obrázek 5.8 – Třídy UML a jejich asociace

Obrázek 5.8 ilustruje další funkci diagramů tříd – schopnost znázornit, kolik objektů se na přidružení podílí. V tomto příkladu je každý konec přidružení označen číslem 1, což znamená, že mezi objekty těchto tříd existuje vztah typu 1:1. To znamená, že každý pacient má přesně jeden záznam a každý záznam uchovává informace právě o jednom pacientovi. Jak je patrné z dalších příkladů, mohou existovat i jiné četnosti. Můžeme definovat, že se na vztahu podílí přesný počet objektů, nebo můžeme symbolem * (viz obrázek 5.9) označit, že se na asociaci účastní neomezený počet objektů.

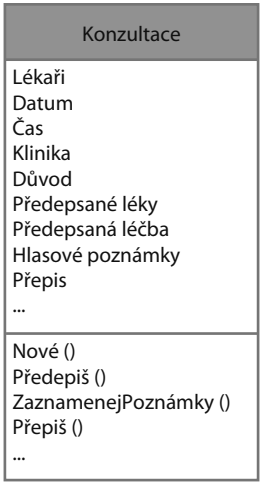
Obrázek 5.9 rozvíjí tento typ diagramu tříd a ukazuje, že objekty třídy Pacient se také podílejí na vztazích s mnoha jinými třídami. Z tohoto příkladu je zřejmé, že přidružení lze pojmenovat, aby čtenáři grafu získali představu o typech existujících vztahů. Jazyk UML rovněž dovoluje specifikovat roli objektů, které se na asociaci podílejí.

Na této úrovni podrobností vypadají diagramy tříd jako sémantické datové modely. Sémantické datové modely se používají při návrhu databází. Znázorňují datové entity, jejich přidružené atributy a vztahy mezi těmito entitami. Tento přístup k modelování poprvé navrhl v polovině 70. let Chen (1976). Od té doby vzniklo několik variant (Codd, 1979; Hammer a McLeod, 1981; Hull a King, 1987), které sdílejí stejnou základní formu.

Jazyk UML nezahrnuje konkrétní notaci pro toto databázové modelování, protože předpokládá objektově orientovaný vývojový proces a modeluje data pomocí objektů a jejich vztahů. Pomocí jazyka UML lze však reprezentovat i sémantický datový model. Entity sémantického datového modelu si můžeme představit jako zjednodušené objektové třídy (nemají žádné operace), atributy jako atributy objektové třídy a vztahy jako pojmenovaná přidružení mezi objektovými třídami.



Obrázek 5.9 – Třídy a asociace v systému MHC-PMS



Obrázek 5.10 – Třída konzultace

Při znázorňování asociací mezi třídami je výhodné reprezentovat tyto třídy co nejjednodušším způsobem. Chceme-li třídy definovat podrobněji, doplníme informace o jejich atributech (vlastnosti objektu) a operacích (činnostech, které od objektu požadujeme). Například objekt Pacient bude mít atribut Adresa a můžeme k němu přidat i operaci označenou ZměňAdresu. Tato operace se bude volat, když pacient oznámí, že se přestěhoval na jinou adresu. V jazyce UML se atributy a operace znázorňují rozšířením jednoduchého obdélníku, který označuje třídu. To je patrné na obrázku 5.10, kde:

1. Název objektové třídy je umístěn v horní části.
2. Atributy třídy jsou ve střední části. Nesmí zde chybět názvy atributů a volitelně mohou být uvedeny jejich typy.
3. Operace (které se v jazyce Java a jiných objektově orientovaných programovacích jazycích nazývají metody) přidružené k objektové třídě jsou v dolní části obdélníka.

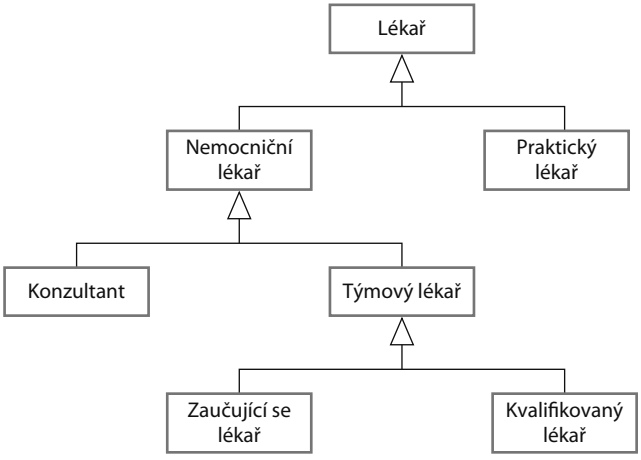
Obrázek 5.10 znázorňuje možné atributy a operace třídy Konzultace. V tomto příkladu předpokládáme, že lékaři nahrávají hlasové poznámky, které jsou později přepisovány za účelem zaznamenání podrobností o konzultaci. Při předepsání léku musí příslušný lékař použít metodu Předepiš, která generuje elektronický recept.

5.3.2 Generalizace

Generalizace je běžná metoda, díky níž zvládáme složitost světa. Místo toho, abychom se seznamovali s podrobnými vlastnostmi každé entity, se kterou se setkáváme, řadíme tyto entity do obecnějších tříd (zvířata, auta, domy atd.) a učíme se vlastnosti těchto tříd. Díky tomu můžeme odvodit, že různí členové těchto tříd mají některé společné vlastnosti (například veverky a krysy jsou hlodavci). Můžeme učinit obecné závěry, které se vztahují na všechny členy třídy (například všichni hlodavci mají zuby, kterými dokážou hlodat).

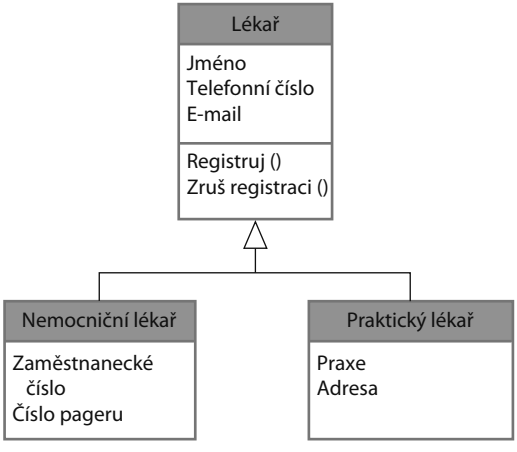
V modelovacích systémech je často užitečné prozkoumat třídy v systému a vyhodnotit, zda neexistuje prostor pro jejich generalizaci. To znamená, že společné informace lze uchovávat jen na jednom místě. Jedná se o vhodný návrhový postup, protože pokud jsou navrženy změny, není nutné kontrolovat

všechny třídy v systému, zda na ně příslušná změna bude mít vliv. V objektově orientovaných jazycích, jako je Java, se generalizace implementuje pomocí mechanismů dědění tříd, které jsou součástí jazyka.



Obrázek 5.11 – Hierarchie generalizace

Jazyk UML má konkrétní typ asociace, který označuje generalizaci (viz obrázek 5.11). Generalizace se znázorňuje pomocí šipky, která směřuje k obecnější třídě. Z toho je zřejmé, že praktické lékaře a nemocniční lékaře lze zobecnit jako lékaře a že existují tři typy nemocničních lékařů – ti, kteří nedávno absolvovali lékařskou fakultu a je nutné na ně dohlížet (zaučující se lékař), ti, kteří mohou pracovat bez dozoru v rámci konzultačního týmu (registrovaný lékař), a konzultanti, což jsou zkušení lékaři s plnými rozhodovacími pravomocemi.



Obrázek 5.12 – Hierarchie generalizace s doplněnými podrobnostmi

Při generalizaci se atributy a operace přidružené ke třídám vyšší úrovně přidružují také ke třídám nižší úrovně. Třídy nižší úrovně jsou v zásadě podtřídami, které dědí atributy a operace od svých nadtříd. Tyto třídy nižší úrovně pak doplňují konkrétnější atributy a operace. Například všichni lékaři mají jméno a telefonní číslo. Všichni nemocniční lékaři mají zaměstnanecké číslo a oddělení, kde jsou zaměstnáni,

ale praktičtí lékaři tyto atributy nemají, protože pracují samostatně. Místo toho však používají název a adresu své praxe. To je patrné na obrázku 5.12, který znázorňuje část hierarchie generalizace, kterou jsme doplnili o atributy tříd. Operace přidružené ke třídě Lékař mají zajistit registraci a zrušení registrace daného lékaře v systému MHC-PMS.

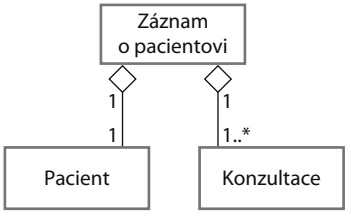
5.3.3 Agregace

Objekty reálného světa se často skládají z různých částí. Například studijní materiály ke kurzu mohou zahrnovat knihu, prezentace PowerPoint, testy a doporučení další literatury. Někdy je potřeba zajistit, aby model systému toto složení dokumentoval. Jazyk UML poskytuje speciální typ přidružení tříd označovaný jako agregace. Znamená, že jeden objekt (celek) se skládá z jiných objektů (částí). Při znázornění používáme kosočtverec vedle třídy, která reprezentuje celek. To je patrné na obrázku 5.13, který znázorňuje, že záznam o pacientovi obsahuje položku Pacient a neomezený počet Konzultací.

5.4 Behaviorální modely

Behaviorální modely znázorňují dynamické chování systému při jeho činnosti. Ukazují, co se děje nebo má dít, když systém reaguje na impulsy ze svého prostředí. Tyto stimuly můžeme zařadit do dvou typů:

1. *Data* – doručována jsou určitá data, která musí systém zpracovat.
2. *Události* – nastávají jisté události, které aktivují systémové zpracování. Události mohou mít přidružena data, ale neplatí to vždy.



Obrázek 5.13 – Přidružení typu agregace

DIAGRAMY DATOVÉHO TOKU

Diagramy datového toku (DFD – data-flow diagram) jsou systémové modely, které znázorňují funkční perspektivu, kde každá transformace reprezentuje jedinou funkci nebo proces. Tyto diagramy umožňují znázornit, jak data postupují sekvencí kroků svého zpracování. Krokem zpracování může být například odfiltrování duplicitních záznamů z databáze zákazníků. Data se v každém kroku transformují a následně postupují do další fáze. Tyto kroky či transformace zpracování reprezentují softwarové procesy či funkce, kdy lze pomocí diagramů datového toku dokumentovat návrh softwaru.

http://www.SoftwareEngineering-9.com/Web/DFDs

Mnoho podnikových systémů patří mezi systémy zpracování dat, které primárně slouží k manipulaci s daty. Jsou řízeny vstupem dat do systému a poměrně v malé míře jsou řízeny zpracováním externích událostí. Zpracování v tomto případě zahrnuje sekvenci akcí s daty a generování výstupu. Například systém účtování telefonních služeb přijímá informace o voláních, která zákazník uskutečnil, počítá cenu těchto volání a generuje fakturu, která bude příslušnému zákazníkovi odeslána. Oproti tomu systémy

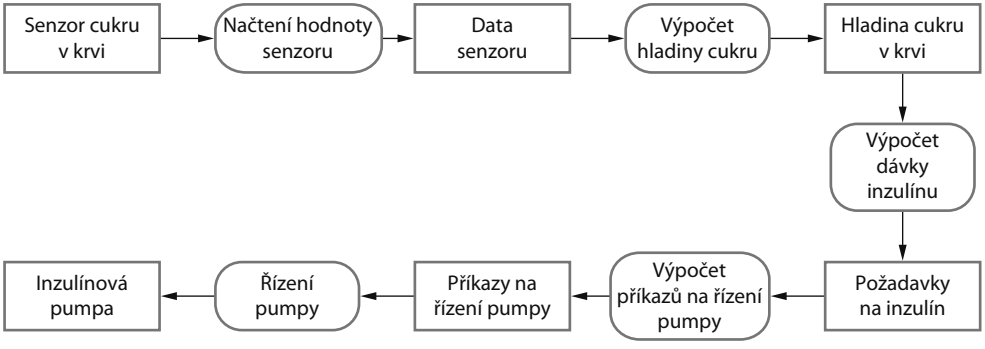
fungující v reálném čase jsou často řízeny událostmi a pouze minimálně zpracovávají data. Například systém ústředny pro pevnou telefonní síť reaguje na různé události – například na zvednuté sluchátko generováním oznamovacího tónu nebo na stisknutí tlačítek telefonu zaznamenáním telefonního čísla.

5.4.1 Modelování řízené daty

Modely řízené daty znázorňují sekvenci akcí, které se podílejí na zpracování vstupních dat a generování příslušného výstupu. Hodí se zejména k analýze požadavků, protože umožňují vizualizovat kompletní postup zpracování v systému. To znamená, že ukazují celou sekvenci akcí, která probíhá od příjmu vstupu po generování odpovídajícího výstupu, který představuje odpověď systému.

Modely řízené daty patřily mezi první grafické softwarové modely. V 70. letech byly v rámci strukturovaných metod, jako je DeMarcova strukturovaná analýza (DeMarco, 1978), zavedeny diagramy datového toku (DFD), které umožňovaly ilustrovat kroky zpracování v systému. Modely datového toku jsou užitečné, protože sledování a dokumentování toho, jak data související s určitým procesem procházejí systémem, pomáhá analytikům a návrhářům porozumět fungování systému. Diagramy datového toku jsou jednoduché a intuitivní a obvykle je lze vysvětlit potenciálním uživatelům systému, kteří se pak mohou účastnit na validaci modelu.

Jazyk UML neposkytuje podporu diagramů datového toku, protože byly původně navrženy a používaly se k modelování zpracování dat. Důvod spočívá v tom, že diagramy datového toku se zaměřují na systémové funkce a nerozpoznávají systémové objekty. Vzhledem k tomu, že systémy řízené daty jsou v podnicích velmi běžné, však jazyk UML 2.0 zavedl diagramy aktivity, které se podobají diagramům datového toku. Například obrázek 5.14 znázorňuje řetězec akcí zpracování, který se uplatňuje u softwaru inzulínové pumpy. Na tomto diagramu jsou patrné kroky zpracování (reprezentované jako aktivity) a data směřující mezi těmito kroky (reprezentovaná jako objekty).



Obrázek 5.14 – Model aktivity fungování inzulínové pumpy

Alternativní způsob znázornění sekvence zpracování v systému využívá sekvenční diagramy UML. Již jsme viděli, jak lze tyto diagramy uplatnit při modelování interakcí. Pokud je však nakreslíme tak, aby zprávy proudily pouze zleva doprava, pak znázorňují sekvenční zpracování dat v systému. To je patrné na obrázku 5.15, kde se používá sekvenční model zpracování objednávky a jejího odeslání dodavateli. Sekvenční modely se zaměřují na objekty v systému, zatímco diagramy datového toku zvýrazňují funkce. Odpovídající diagram datového toku pro zpracování objednávky je k dispozici na webových stránkách knihy.