KAPITOLA 8

Jak správně pochopit uživatelské požadavky

Systém pro evidenci chemikálií právě prodělával svůj první požadavkový workshop, na kterém se mělo zjistit, co od něj potřebují chemici. Mezi účastníky workshopu byla Lori, systémová analytička a zároveň moderátorka; Tim, produktový šampión chemiků; Sandy a Peter, další dva zástupci chemiků, a hlavní vývojářka Helen. Lori workshop otevřela tím, že všem poděkovala za účast, a pak se rovnou vrhla na věc.

„Tim, Sandy a Peter našli asi patnáct případů užití, které budou chemici od systému potřebovat," řekla všem. „Workshopy děláme kvůli tomu, abychom si tyto případy užití prošli a věděli, jaké funkce budou v systému vyžadovat. Jako nejdůležitější jste označili případ jménem Objednávka chemikálie, pro který už Tim napsal stručný popis, takže jím začneme. Time, jak si v systému představuješ objednávání chemikálií?"

„Za prvé," řekl Tim, „libovolné chemikálie si můžou objednávat jen ti, kteří mají povolení od vedoucího své laboratoře."

„Dobře, to zní jako podnikatelské pravidlo," odpověděla Lori. „Nejspíš narazíme i na další, takže na ně tady na tabuli nachystám samostatnou stránku." Kromě toho Lori nalistovala případ užití Objednávka chemikálie a mezi vstupní podmínky připsala „uživatel je autentizován" a „uživatel má právo objednávat chemikálie". „Má objednávka ještě nějaké další předpoklady?"

„Ještě než si něco objednám přímo od dodavatele, chtěla bych vědět, jestli už to nemáme ve skladu," řekla Sandy. „Takže by skladová databáze měla být při vyplňování žádosti online, abych zbytečně neztrácela čas."

Lori přidala novou vstupní podmínku. V další půlhodině pak skupinu provedla diskusemi o tom, jak si představují objednávku nové chemikálie. Na několik stránek papírové tabule zapsala všechny informace o vstupních i výstupních podmínkách a jednotlivých dialozích mezi uživatelem a systémem. Zeptala se uživatelů, jak by

se případ užití lišil, kdyby si namísto ze skladu objednávali přímo od dodavatele. Během půl hodiny měla celá skupina slušnou představu, jak by si uživatelé mohli objednávat chemikálie, a přesunula se na další případ užití.

Softwarové systémy mají lidé od toho, aby s nimi udělali něco užitečného, a slibný sklon k návrhu čím dál použitelnějších programů vykazuje celý softwarový průmysl (Constantine a Lockwoodová 1999, Nielsen 2000). Zásadním předpokladem pro návrh použitelného softwaru je vědět, co s ním uživatelé hodlají dělat.

Pro sbírání uživatelských požadavků se řadu let používaly takzvané scénaře užití (McGraw a Harbison 1997). *Scénář* je popis jednoho konkrétního použití systému. Ivar Jacobson s kolektivem (1992), Larry Constantine a Lucy Lockwoodová (1999), Alistair Cockburn (2001) a další formalizovali uživatelský přístup k analýze požadavků do takzvaných *případů užití*. Pomocí případů užití se dají dobře zjistit požadavky na podnikatelské systémy, webové stránky, služby poskytované jedním systémem jinému a systémy pro ovládání hardwaru. Dávkové procesy, systémy pro náročné výpočty, systémy pro ukládání dat a další podobné systémy mívají jen několik málo jednoduchých případů užití. Jejich složitost spočívá v prováděných výpočtech nebo analýze dat, nikoliv v interakci s uživatelem. Při analýze požadavků real-time systémů se často používá seznam vnějších podnětů a odpovídajících reakcí systému. Tato kapitola popisuje, jak se používají případy užití a tabulky událostí a odpovědí.

Upozornění

Nesnažte se každý požadavek, na který narazíte, dostat do podoby případu užití. Případy užití odhalí většinu funkčních požadavků, ale ne nutně všechny.

Případy užití

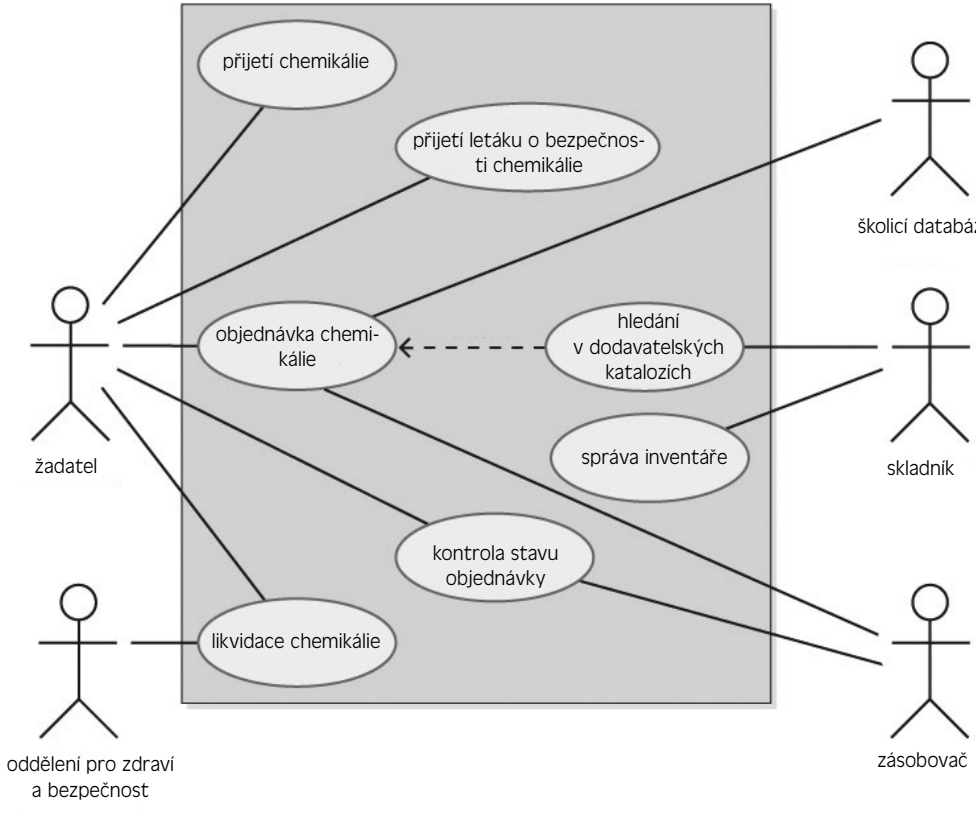
Případ užití popisuje posloupnost interakcí mezi systémem a vnějším aktérem. *Aktér* je osoba, jiný softwarový systém nebo hardwarové zařízení, které systém používá k dosažení nějakého užitečného cíle (Cockburn 2001). Místo slova aktér se někdy používá výraz uživatelská role, protože aktéři jsou vlastně role hrané členy jedné nebo více uživatelských tříd (Constantine a Lockwoodová 1999). Například v Systému pro evidenci chemikálií existuje případ užití *Objednávka chemikálie*, ve kterém hraje roli aktér jménem Žadatel. Všimněte si ale, že v tomto systému neexistuje žádná uživatelská třída jménem Žadatel. Chemikálie si mohou objednávat chemici i skladníci, takže roli Žadatel může hrát libovolná z těchto uživatelských tříd.

Případy užití pochází z objektově orientovaného vývoje softwaru, ale dají se použít u projektů navrhovaných libovolnou metodologií. Případy užití také tvoří základ rozšířené metodologie *Unified Software Development Process* (Jacobson, Booch a Rumbaugh 1999).

Případy užití poněkud mění perspektivu sbírání požadavků a snaží se zjistit, čeho potřebují dosáhnout *uživatelé* – tradiční přístup se naproti tomu snažil zjistit, co by podle názoru uživatelů měl dělat *systém*. Cílem případů užití je popsat všechny úkoly, které uživatelé prostřednictvím systému potřebují udělat. Účastníci projektu se musí ujistit, že každý případ užití spadá do vymezeného rozsahu, a teprve pak jej mohou přijmout do směrné verze dokumentace požadavků. Výsledná skupina případů užití by teoreticky měla popisovat veškerou požadova-

nou funkcionalitu systému. V praxi dosáhnete úplného pokrytí jen zřídka, ale případy užití vás alespoň dostanou dál, než libovolná z ostatních technik, které jsem kdy použil.

Pro obecný grafický zápis uživatelských požadavků se používají *diagramy případů užití*. Na obrázku 8.1 je například vidět diagram případu užití ze Systému pro evidenci chemikálií. Použité symboly odpovídají UML (Booch, Rumbaugh a Jacobson 1999; Armour a Miller 2001): velký obdélník představuje hranice systému a čáry spojují panáčky neboli aktéry s případy užití (elipsami). Všimněte si, jak se tento diagram podobá kontextovému diagramu z obrázku 5.3. U diagramu případů užití slouží velký obdélník k oddělení obecných vnitřních částí systému (případů užití) a vnějších aktérů. I na kontextovém diagramu jsou vidět objekty, které leží za hranicemi systému, ale vnitřek systému je u něj skrytý.



Obrázek 8.1. Systém pro evidenci chemikálií, část diagramu případů užití.

Případ užití versus scénář užití

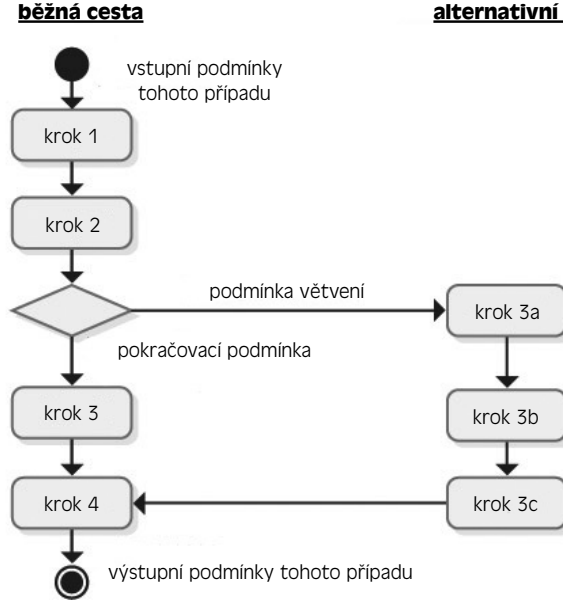
Případ užití je samostatná činnost, kterou může aktér provést a získat tak hodnotný výsledek. Jeden případ užití může zahrnovat několik podobných úkolů se společným cílem. Případ užití je skupina navzájem souvisejících scénářů užití, a scénář užití je jeden konkrétní průchod případem užití. Při analýze uživatelských požadavků můžete vyjít z abstraktního případu

užití a dopsat k němu konkrétní scénáře, a nebo můžete naopak konkrétní scénáře zobecnit do abstraktnějšího případu užití. Podrobnou šablonu pro popis případů užití uvidíte později v této kapitole. V popisu případu užití jsou nejdůležitější následující informace:

- Jedinečný identifikátor.
- Stručný název uživatelského úkolu, například Podání objednávky.
- Krátký textový popis úkolu v přirozeném jazyce.
- Vstupní podmínky, které musí být splněné před začátkem případu.
- Výstupní podmínky, které popisují stav systému po úspěšném skončení případu.
- Číslovaný seznam kroků, na kterých je vidět posloupnost dialogů nebo interakcí mezi aktérem a systémem, a jež vedou od vstupních podmínek k výstupním.

Jeden scénář se označí jako *běžná cesta*; říká se mu též hlavní cesta, základní cesta, šťastná cesta, hlavní scénář nebo úspěšný scénář. Běžnou cestou skrz případ „Objednávka chemikálie" je objednávka chemikálie, která je na skladě.

Další možné scénáře se označují jako *alternativní cesty* nebo též *sekundární scénáře* (Schneider a Winters 1998). Alternativní cesty představují různé variace běžného postupu, a stejně jako on vedou k úspěšnému splnění úkolu a dodrží výstupní podmínky případu. Běžná cesta se může v nějakém rozhodovacím místě rozvětvit na několik alternativních cest, které se o něco později zase sloučí dohromady. Většina případů užití se dá popsat obyčejným textem, ale pro zápis složitějších případů je lepší vývojový diagram nebo diagram aktivit podle UML (viz obrázek 8.2). Na vývojových diagramech a diagramech aktivit jsou dobře vidět rozhodovací místa a podmínky, které způsobí rozvětvení běžné cesty do několika alternativních.

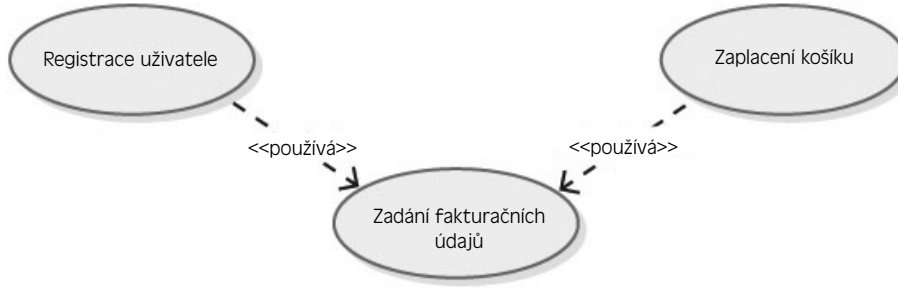


Obrázek 8.2. Běžné a alternativní cesty skrz jeden případ užití. Použitý zápis odpovídá diagramům aktivit podle UML.

Alternativní cestou skrz Objednávku chemikálie je objednávka chemikálie od dodavatele. Aktérův cíl je v obou případech tentýž (chce si objednat chemikálii), a proto oba scénáře patří do jednoho případu užití. Některé části alternativní cesty budou stejné jako v běžném případě, jinde se alternativní cesta odchýlí. V našem případě může uživatel žádanou látku v rámci alternativní cesty hledat v dodavatelských katalozích. Pokud je alternativní cesta také samostatný případ užití, můžete běžnou cestu takzvaně *rozšířit* a tento případ užití do ní vložit (Armour a Miller 2001). Rozšiřování případů užití je vidět na diagramu 8.1, ve kterém se Objednávka chemikálie rozšiřuje o případ „Hledání v dodavatelských katalozích". Hledání v dodavatelských katalozích je samostatný případ užití, protože skladníci v nich potřebují hledat i bez objednávání.

Někdy má několik případů užití společnou skupinu kroků. Abyste tyto kroky nemuseli v každém případu užití opakovat, můžete z nich vytvořit samostatný případ užití, který do ostatních případů *vložíte*. Je to jako když v programovacím jazyce voláte podprogram.

Představte si například elektronický obchod, ve kterém si zákazník vybere nějaké zboží a přejde k virtuální pokladně. Pokud už je registrovaný, stačí se přihlásit a systém si potřebné fakturační údaje vybere z informací, které uživatel zadával při registraci. Pokud registrovaný není, systém ho do registrace nebude nutit a na fakturační údaje se ho v rychlosti zeptá. Zadávání fakturačních údajů by se tedy mohlo zapsat jako samostatný případ užití, který se vloží do případů Registrace uživatele a Zaplacení košíku.



Obrázek 8.3. Elektronický obchod a vkládání případů užití.

Upozornění

Neztrácejte čas zbytečnými diskusemi o tom, jestli, kdy a jak se vkládání a rozšiřování případů hodí používat. V praxi se nejčastěji používá osamostatnění společných kroků do vkládaného případu užití.

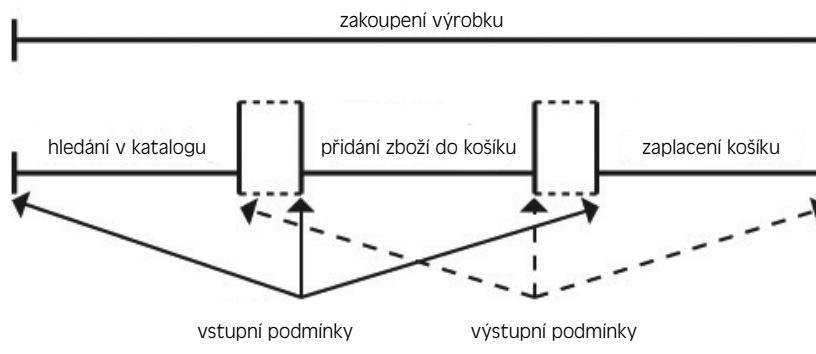
Situace, ve kterých se úkol nedá dokončit, se označují jako *výjimky*. Jednou z výjimek případu Objednávka chemikálie například je, že „chemikálie není komerčně dostupná". V rámci sbírání požadavků byste měli popsat i zpracování výjimek, jinak se můžete dočkat jednoho z následujících výsledků:

1. Vývojáři si způsob zpracování výjimek domyslí.
2. Systém se při výskytu výjimky složí, protože na výjimku nikdo nepomyslel.

Dá se celkem bezpečně předpokládat, že pády systému mezi uživatelskými požadavky nenajdete.

Výjimky se někdy berou jako zvláštní druh alternativních cest (Cockburn 2001), ale praktičtější je, když je pojmete samostatně. Alternativní cesty totiž nemusíte implementovat všechny (případně si některé můžete nechat na pozdější verzi), zatímco výjimky implementovat musíte, protože brání úspěšnému dokončení některých scénářů. Každý programátor ví, že zpracování výjimek často spolkne největší část práce. Řada chyb v hotových systémech spočívá právě ve zpracování výjimek (a nebo jeho absenci). Popisem výjimek už během sbírání požadavků zvyšujete spolehlivost systému.

U mnoha systémů mohou uživatelé z několika menších případů užití sestavit jeden větší, který popisuje nějaký složitější úkol. Mezi případy užití elektronického webového obchodu by mohlo patřit například Hledání v katalogu, Přidání zboží do košíku nebo Zaplacení košíku. Pokud se libovolný z těchto úkolů dá provést samostatně, jde o samostatné případy užití. Všechny tři se ale dají projít také společně, v rámci většího případu nazvaného Zakoupení výrobku (viz obrázek 8.4). Aby to fungovalo, po skončení každého z jednotlivých případů musí být systém v takovém stavu, aby se dalo okamžitě začít s následujícím případem užití. Jinými slovy, musí výstupní podmínky předchozího případu užití splňovat vstupní podmínky toho následujícího. Podobná je situace u transakčních systémů (například bankomatů), u kterých musí každý případ užití končit stavem, jenž umožňuje začátek následujícího. Vstupní a výstupní podmínky všech transakcí si musí odpovídat.



Obrázek 8.4. Vstupní a výstupní podmínky definují hranice jednotlivých případů užití, které se dají spojit do většího celku.

Jak se případy užití hledají

Případy užití se dají hledat několika způsoby (Ham 1998, Larman 1998):

- Najdete aktéry a pak všechny podnikatelské procesy, kterých se tito aktéři účastní.
- Najdete vnější události, na které systém musí odpovídat, a tyto události pak spojíte s příslušnými aktéry a případy užití.
- Pomocí scénářů užití popíšete všechny podnikatelské procesy, scénáře užití zobecníte do případů užití a nakonec najdete aktéry, kterých se tyto případy užití týkají.

- Případy užití odvodíte ze stávajících funkčních požadavků. Pokud některé z funkčních požadavků nevedou k žádnému případu užití, zamyslete se, jestli je doopravdy potřebujete.

Systém pro evidenci chemikálií se vydal první cestou. Analytici uspořádali sérii dvouhodinových až tříhodinových případových workshopů, které se konaly zhruba dvakrát týdně. Každá třída uživatelů měla svůj samostatný workshop, který probíhal zároveň s workshopy pro ostatní třídy. Tento systém se osvědčil, protože společných případů užití měli uživatelé jen málo. Každého workshopu se účastnil produktový šampión za danou třídu, několik dalších vybraných zástupců uživatelů a vývojář. Vyvojáři si díky účasti na workshopech udělají lepší představu o systému, na kterém budou pracovat. Zároveň také poslouží jako spojení s realitou, kdyby některý z navrhovaných požadavků obsahoval něco prakticky neproveditelného.

Ještě před začátkem workshopů se každý analytik uživatelů zeptal na úkoly, které se systémem budou chtít dělat. Každý z těchto úkolů se stal kandidátem na případ užití. U několika navrhovaných případů užití se usoudilo, že spadají mimo rozsah projektu, a tak byly vyřazeny. Během další analýzy skupina zjistila, že několik z dalších návrhů jsou ve skutečnosti jen scénáře, které se dají spojit do jednoho obecnějšího případu užití. Skupina také přišla na několik dalších případů užití, které ze začátku vynechala.

Několik uživatelů přišlo s případy užití, které nebyly formulovány jako úkol (například „Leták o bezpečnosti chemické látky"). Název případu užití by ale měl vystihovat úkol, kterého chce uživatel dosáhnout, takže by se vám víc hodilo sloveso. Co chce uživatel s letákem o bezpečnosti dělat? Chce si ho prohlédnout, objednat, poslat e-mailem, opravit v něm chyby, smazat ho, vytvořit? Někdy zase uživatel přijde s názvem, který je ve skutečnosti jen částí příslušného případu užití (například „skenování čárového kódu"). Analytik potřebuje vědět, čeho se uživatel skenováním čárového kódu snaží dosáhnout. Může se zeptat: „A když skenujete čárový kód z toho obalu, proč to vlastně děláte?" Dejme tomu, že uživatel odpoví: „Musím čárový kód naskenovat, abych tu chemikálii mohl zapsat do inventáře své laboratoře." V tom případě by se případ užití mohl jmenovat třeba Zapsání chemikálie. Skenování čárového kódu je pouze jeden krok v interakci mezi aktérem a systémem, který chemikálii zapisuje do laboratorního inventáře.

Uživatele většinou jako první napadnou ty nejdůležitější případy užití, takže pořadí nalezených případů leccos naznačuje o jejich důležitosti. Při rozdělování priorit také můžete postupovat tak, že ke každému navrhovanému případu užití napíšete stručný popis. Pak si tyto kandidáty rozdělte podle priority a podle verzí systému, ve kterých je chcete implementovat. Ty nejdůležitější případy užití byste měli podrobně popsat jako první, aby se vývojáři mohli pustit do jejich implementace.

Dokumentace případů užití

V této fázi by se účastníci měli soustředit na takzvané *principiální případy užití*. Constantine a Lockwoodová definují principiální případ užití jako „zjednodušený, zobecněný, netechnický a implementačně nezávislý popis jednoho úkolu nebo interakce, který vystihuje podstatu této interakce". Jinými slovy byste se měli soustředit na cíl, kterého se uživatel snaží dosáhnout, a systémové funkce, jež mu v tom budou pomáhat. Principiální případy užití jsou obecnější než takzvané *konkrétní případy užití*, které interakci se systémem popisují podrobně. Aby byl

tento rozdíl o něco jasnější, podívejte se na následující dva způsoby, jak popsat potenciální začátek objednávky nějaké chemikálie:

- Konkrétní případ užití: *Zadejte číselný kód chemikálie.*
- Principiální případ užití: *Zadejte, o jakou chemikálii máte zájem.*

Principiální případ užití je formulovaný tak, aby neomezoval možné implementace uživatelského požadavku. Uživatel může zadat číselný kód, načíst chemickou strukturu ze souboru, nakreslit schéma myší, vybrat chemikálii ze seznamu nebo cokoliv dalšího. Když se příliš brzy pustíte do podrobného popisu interakce mezi uživatelem a systémem, zbytečně omezujete myšlenkové pochody účastníků workshopu. Díky své nezávislosti na implementaci se navíc principiální případy užití dají lépe recyklovat v pozdější verzi nebo na jiném projektu.

Účastníci workshopů zabývajících se Systémem pro evidenci chemikálií začali tím, že napsali stručný popis každého z případů užití a našli jeho aktéry. Pak definovali vstupní a výstupní podmínky, které tvoří hranici každého případu (mezi nimiž se odehrávají všechny jejich kroky). Další přišel na řadu odhad četnosti, s jakou se případ bude vyskytovat. Podle této četnosti se dá včas posoudit, kolik uživatelů bude případ používat souběžně, a jaké budou jeho nároky na kapacitu. Pak se analytik účastníků zeptal, jak by si při jednotlivých případech užití představovali interakci se systémem. Výsledná posloupnost uživatelských akcí a odpovědí systému tvoří běžnou cestu skrz případ užití; jednotlivé kroky jsou pro větší přehlednost číslované. Budoucí uživatelské rozhraní a konkrétní nástroje pro práci se systémem si sice každý z účastníků představoval jinak, ale na principech dialogu mezi aktéry a systémem se skupina shodla.

Z praxe

Jak zůstat na hřišti

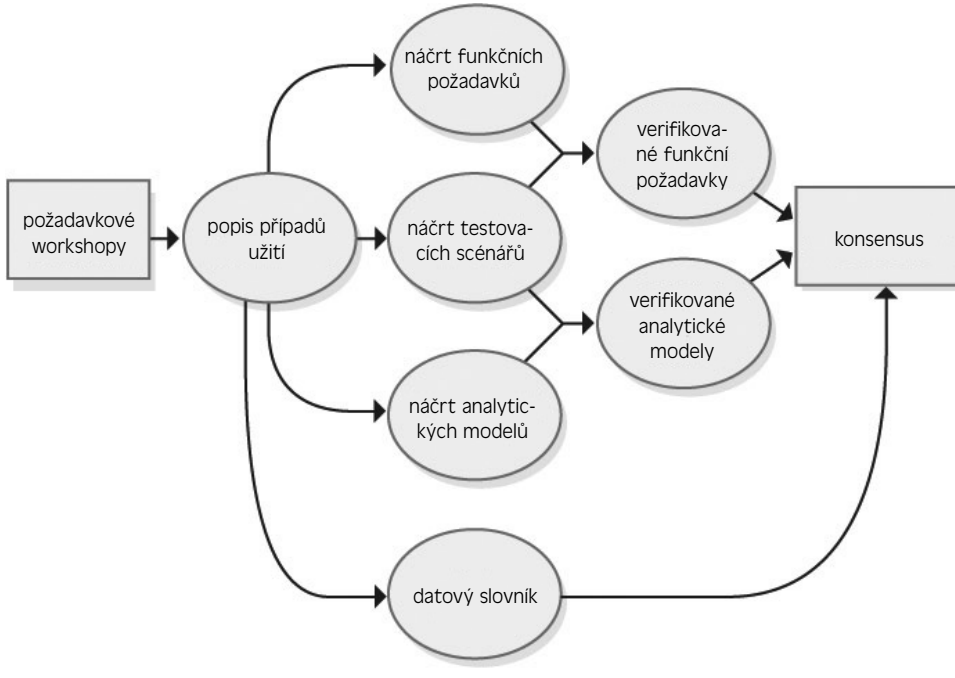
Při prohlížení osmi kroků nějakého případu užití jsem si jednou uvědomil, že jeho výstupní podmínky byly splněné už po pátém z nich. Kroky 6, 7 a 8 tedy byly zbytečné, ležely za hranicí případu. Podobně si musíte dát pozor, aby byly před prvním krokem splněné všechny vstupní podmínky. Kdykoliv si čtete popis případu užití, podívejte se, jestli je skutečně ohraničený vstupními a výstupními podmínkami.

Analytik si akce jednotlivých uživatelů a odpovědi systému poznačil na lístečky, které pak nalepil na papírovou tabuli. Další možnost je promítnout šablonu případu užití z počítače na velké plátno a během diskuse ji doplnit, ale to někdy zdržuje.

Podobně vypadalo i hledání alternativních cest a výjimek. Uživatel, který říká „za normálních okolností…," popisuje běžnou cestu skrz případ užití. Věty typu „ale také by mělo jít…" naznačují spíše alternativní cestu. Na hodně výjimek se přišlo, když se analytik zeptal například: „Co by se mělo stát, když ta databáze momentálně nebude dostupná?" Nebo: „Co když se tahle látka momentálně neprodává?" Tyto workshopy jsou mimo jiné dobrá příležitost, abyste s uživateli probrali jejich požadavky na kvalitu systému, například dobu odezvy, dostupnost a spolehlivost, omezení návrhu rozhraní nebo bezpečnostní požadavky.

Když už nikoho nenapadaly žádné další alternativní cesty, výjimky ani speciální požadavky, diskuse se přesunula na další případ užití. Účastníci se nesnažili o nějaký maratón, ve kterém by popsali všechny případy použití najednou, ani se případy nesnažili popsat do posledního detailu. Shodli se na tom, že budou případy vylepšovat postupně.

Na obrázku 8.5 je vidět celý systém práce na případech užití. Po skončení workshopů napsal analytik ke každému případu užití dokumentaci podobnou té, která je vidět na obrázku 8.6. Srdcem celého případu je interakce mezi aktéry a systémem, a tu lze zapsat dvěma způsoby. Na obrázku 8.6 má tento dialog podobu číslovaných kroků, u kterých je napsáno, jaká z obou stran (systém nebo konkrétní aktér) příslušný krok dělá. Tentýž zápis se používá i pro alternativní cesty a výjimky, u nichž je navíc vidět, ve kterém kroku běžné cesty se mohou objevit. Druhá možnost je zapsat dialog jako tabulku se dvěma sloupci, viz obrázek 8.7 (Wirfs-Brock 1993). Akce aktérů jsou v levém sloupci, odpovědi systému v pravém, a pořadí událostí je dané číslováním. Tento způsob funguje dobře tehdy, když se systémem pracuje pouze jeden aktér. Čitelnost můžete zlepšit tím, že každou z akcí nebo odpovědí uvedete na samostatnou řádku, aby se střídaly – viz obrázek 8.8.



Obrázek 8.5. Systém práce na případech užití.

Případ užití číslo:	**PU–1**	Název:	**Objednání chemikálie**
Autor:	Tim	Autor posledních změn:	Janice
Datum vytvoření:	12. 4. 02	Datum poslední změny:	27. 12. 02
Aktéři	Objednavatel		
Popis	Objednavatel zadá, o jakou látku má zájem. Zadat ji může jmé-nem, jejím číslem nebo načtením její struktury z aplikace pro kreslení chemických struktur. Systém objednávku vyřeší tak, že Objednavateli nabídne nový nebo použitý kontejner s chemikálií ze skladu a nebo ho nechá objednat přímo od externího dodava-tele.		
Vstupní podmínky	1. Uživatel musí být autentizován. 2. Uživatel musí mít právo objednávat chemikálie. 3. Databáze chemického skladu je online.		

Výstupní pod-mínky	1. Objednávka se uloží v Systému pro evidenci chemikálií. 2. Objednávka se e-mailem odešle do skladu nebo Zásobovači.
Běžná cesta	**1.0 Objednávka chemikálie ze skladu** 1. Objednavatel určí, o jakou látku má zájem. 2. Systém ověří, že je zadání v pořádku. 3. Systém vypíše seznam kontejnerů s danou chemikálií, které jsou právě na skladě. 4. Objednavatel si může zobrazit historii použití libovolného kontejneru (viz Zobrazení historie kontejneru). 5. Objednavatel si vybere konkrétní kontejner nebo systém po-žádá o objednání u dodavatele (alternativní cesta 1.1). 6. Objednavatel zadá další informace, které jsou potřeba k do-končení objednávky. 7. Systém objednávku uloží a pošle e-mail do skladu.
Alternativní cesty	**1.1 Objednání chemikálie od dodavatele (z kroku 5)** 1. Objednavatel si látku vyhledá v katalozích dodavatelů. 2. Systém zobrazí seznam dodavatelů s dostupnými velikostmi kontejnerů, jakostí a cenami. 3. Objednavatel si vybere dodavatele, velikost kontejneru, jakost a počet kusů. 4. Objednavatel vyplní další informace, které jsou potřeba k do-končení objednávky. 5. Systém objednávku uloží a pošle e-mail do skladu.

Výjimky	**1.0. V.1 Chybně zadaná chemikálie (z kroku 2)** 1. Systém zobrazí hlášení, že taková chemikálie neexistuje. 2. Systém se objednavatele zeptá, jestli chce zadat jinou chemiká-lii, nebo skončit. 3a. Objednavatel chce zadat jinou chemikálii. 4a. Systém spustí od začátku běžnou cestu. 3b. Objednavatel chce skončit. 4b. Případ užití končí. **1.0.V.2 Chemikálie není komerčně dostupná (z kroku 5)** 1. Systém zobrazí hlášení, že tuto chemikálii nemá v katalogu žádný z dodavatelů. 2. Systém se Objednavatele zeptá, jestli chce objednat jinou che-mikálii, nebo skončit. 3a. Objednavatel chce objednat jinou chemikálii. 4a. Systém spustí od začátku běžnou cestu. 3b. Objednavatel chce skončit. 4b. Případ užití končí.
Vkládané případy užití	**PU–22** Zobrazení historie kontejneru.
Priorita	Vysoká
Četnost užití	Zhruba pětkrát do týdne každým z chemiků, stokrát týdně kaž-dým ze skladníků.
Podnikatelská pravidla	**PP–28** Jakékoliv chemikálie si smí objednávat pouze ti, kteří mají povolení od vedoucího své laboratoře.
Zvláštní požadavky	Systém musí být schopen importovat chemického struktury ve formátu, který používají podporo-vané aplikace pro kreslení che-mických struktur.
Předpoklady	Importované chemické struktury jsou v pořádku.
Poznámky a problémy	Tim musí zjistit, jestli je povolení od správce laboratoře potřeba i při objednávání látek první třídy nebezpečnosti. Termín: 4. 1. 03.

Obrázek 8.6. Částečný popis příkladu Objednání chemikálie.

Uživatelská akce	*Odpověď systému*
1. Zadání chemikálie. 4. Pokud uživatel chce, může si zobrazit historii libovolného kontejneru. 5. Výběr konkrétního kontejneru (→ konec případu) nebo žádost o objednávku u dodavatele (→ alternativní cesta 1.1).	2. Ověření zápisu chemikálie. 3. Zobrazení seznamu všech kontejnerů s danou chemikálií, které jsou na skladu.

Obrázek 8.7. Dvousloupcový popis jednoho dialogu z případu užití.

Uživatelská akce	*Odpověď systému*
1. Zadání chemikálie.	
	2. Ověření zápisu chemikálie.
	3. Zobrazení seznamu všech kontejnerů s danou chemikálií, které jsou na skladu.
4. Pokud uživatel chce, může si zobrazit historii libovolného kontejneru.	
5. Výběr konkrétního kontejneru (→ konec případu) nebo žádost o objednávku u dodavatele (→ alternativní cesta 1.1).	

Obrázek 8.8. Alternativní zápis téhož dialogu.

V souvislosti s případy užití se často přijde na další informace nebo požadavky, které se do žádné ze standardních kolonek šablony nehodí. Pro různé kvalitativní parametry, výkonnostní požadavky a podobné informace můžete založit kolonku Zvláštní požadavky. Nezapomeňte popsat děje, které nejsou z pohledu uživatele vidět – například když musí systém v rámci dokončení případu komunikovat s jiným systémem.

Podrobný popis případu užití není nutný za všech okolností. Alistair Cockburn (2001) rozděluje šablony případů užití na *formální* a *neformální*. Šablona z obrázku 8.6 je podle tohoto rozdělení formální. Neformální případ užití je obyčejný textový popis úkolu, který chce uživatel splnit, a uživatelské interakce se systémem – třeba jen obsah kolonky Popis z obrázku 8.6. Příkladem neformálního případu užití jsou na kartičkách zapsané uživatelské případy, neboli *user stories* z extrémního programování (Jeffries, Anderson a Hendrickson 2001). Formální popis případů užití se hodí v následujících případech:

- Zástupci vašich uživatelů nespolupracují dostatečně těsně s vývojáři.
- Aplikace je složitá a selhání systému s sebou nese velká rizika.
- Vývojáři už žádné podrobnější požadavky než případy užití nedostanou.
- Podle případů užití chcete psát podrobné testovací scénáře.
- Na projektu spolupracuje více týmů na různých místech, takže potřebujete nějakou společnou paměť.

Upozornění

K popisování případů užití nepřistupujte zbytečně dogmaticky. Podstatný je cíl: pochopit uživatelské požadavky na systém natolik dobře, aby podle nich mohli vývojáři pracovat a nemuseli se bát, že budou hotovou práci předělávat.

Na obrázku 8.5 je vidět, že analytici Systému pro evidenci chemikálií po skončení každého workshopu prošli popisy případů užití a napsali podle nich funkční požadavky. (Podrobněji se o tom budeme bavit v následující části této kapitoly.) Pro některé ze složitějších případů užití navíc nakreslili analytické modely, například přechodový diagram, na kterém jsou vidět všechny možné stavy objednávky a povolené přechody mezi nimi.

Další informace

Několik analytických modelů Systému pro evidenci chemikálií najdete v kapitole 11, nazvané *Jeden obrázek vydá za 1 024 slov*.

Jeden nebo dva dny po každém z workshopů předal analytik popsané případy užití a funkční požadavky účastníkům, aby si je mohli prohlédnout před dalším workshopem. Při těchto neformálních revizích se přišlo na řadu chyb: na dříve opomíjené alternativní cesty, nové výjimky, chybné funkční požadavky nebo vynechané dialogy. Mezi workshopy nechávejte alespoň jednodenní mezeru, z odstupu jednoho nebo dvou dnů se lidé na svou práci dívají z jiné perspektivy. Jeden analytik své workshopy pořádal každý den a zjistil, že účastníci mají s hledáním chyb v dokumentaci potíže, protože mají příslušné informace ještě v živé paměti. V podstatě si v hlavě jen přehráli diskusi, která se odehrála na workshopu, a chyby přehlédli.

Upozornění

Zpětnou vazbu od uživatelů, vývojářů a ostatních účastníků si vyžádejte co nejdříve. Nečekejte, až bude sbírání požadavků hotové.

Na samotném začátku vývoje požadavků navrhl hlavní tester podle případů užití konceptuální testovací scénáře, které byly nezávislé na konkrétní implementaci (Collard 1999). Díky těmto testovacím scénářům měl tým lepší představu o tom, jak by se měl systém během různých scénářů užití chovat. A požadavkový analytik si podle nich může ověřit, jestli skutečně našel všechny funkční požadavky, které jsou pro průchod jednotlivými případy užití potřeba. Během závěrečného workshopu si testovací scénáře všichni účastníci společně prošli, aby byla jistota, že si případy užití představují stejně. Stejně jako u jakéhokoliv jiného nástroje pro řízení kvality samozřejmě našli několik dalších chyb v požadavcích i testovacích scénářích.

Další informace

Psaní testovacích scénářů podle požadavků se podrobněji věnuje kapitola 15, nazvaná *Kontrola požadavků*.

Případy užití a funkční požadavky

Vývojáři softwaru nepracují podle podnikatelských požadavků, ani podle případů užití. Pracují podle funkčních požadavků – podle popisu chování různých malých kousků systému, díky kterým systém dokáže provést požadované případy užití a splnit cíle uživatelů. Případ užití popisuje chování systému z pohledu aktéra, a řadu detailů tím pádem vynechává. Pro správný návrh a implementaci vývojář potřebuje i pohledy z jiných stran.

Podle některých odborníků se dají i případy užití považovat za funkční požadavky. V praxi jsem ale viděl problémy způsobené právě tím, že vývojáři dostali do rukou pouze případy užití. Případy užití popisují uživatelův pohled na systém, tedy chování systému při pohledu zvenčí. Neobsahují všechny informace, které při psaní softwaru potřebuje vývojář. Například uživatelé bankomatu se nestarají o žádné operace, které probíhají za oponou (jako je třeba komunikace s počítačem banky). Tyto detaily jsou pro uživatele neviditelné, ale vývojář o nich musí vědět. V případu užití je samozřejmě popsat můžete, ale jen zřídka se k nim dostanete při rozhovoru s uživateli. Když mají vývojáři k dispozici jen případy užití (byť třeba plně formálně popsané), musí se na řadu věcí ptát. Pokud jim chcete usnadnit práci, doporučuji vám, abyste analytika nechali vypracovat podrobné funkční požadavky potřebné pro implementaci každého z případů užití (Arlow 1998).

Hodně funkčních požadavků vypadne přímo z dialogů, které se podle případu užití mají mezi aktérem a systémem odehrát. Některé z nich jsou zjevné, například: „Každému novému požadavku systém přiřadí jedinečné číslo." Pokud jsou tyto podrobnosti naprosto jasně vidět už z případu užití, ve specifikaci už je opakovat nemusíte. Jiné funkční požadavky se v popisu případu užití neobjevují – analytik je musí odvodit na základě svého pochopení případu a provozního prostředí systému. Tento překlad mezi uživatelským a vývojářským pohledem na požadavky je jedním z mnoha případů, kdy analytik zvyšuje hodnotu projektu.

U Systému pro evidenci chemikálií se případy užití používaly především pro nalezení potřebných funkčních požadavků. Jednodušší případy užití analytici popisovali jen neformálně. Z hotových případů užití pak odvodili všechny funkční požadavky, po jejichž implementaci měl být systém schopen provést daný případ užití včetně všech alternativních cest a ošetření výjimek. Funkční požadavky analytici popsali ve specifikaci požadavků, která byla uspořádána podle funkcí systému.

Dokumentace funkčních požadavků spojených s každým případem užití se dá provést několika způsoby. Záleží především na tom, jestli váš tým bude navrhovat, implementovat a testovat podle případů užití, podle specifikace, a nebo podle obojího. Žádná z následujících metod není úplně bez chyby, takže si jednoduše vyberte přístup, který nejlépe odpovídá vašim potřebám ohledně dokumentace a správy požadavků.

Pouze případy užití

Jedna možnost je vložit popis funkčních požadavků přímo do případů užití. Samostatné specifikaci požadavků se tím nevyhnete – budete ji potřebovat pro parametrické požadavky a případně i pro funkční požadavky, které nejsou spojené s žádným případem užití. Někdy se stává, že jeden funkční požadavek potřebujete vložit do několika různých případů užití. (Pokud například pět případů užití vyžaduje autentizaci uživatele, nebudete kvůli tomu psát pět různých bloků kódu.) V takových případech funkční požadavky nezdvojujte, napište příslušný požadavek jen jednou a do všech případů užití ho vložte pomocí odkazu. V některých případech se tento problém dá vyřešit také vkládáním celých případů užití, o kterém jsme se bavili dříve v této kapitole.

Kombinace případů užití a specifikace požadavků

Druhá možnost je popsat případy užití jen stručně a odvodit z nich podrobné funkční požadavky. V tomto případě potřebujete, aby se navzájem související případy užití a funkční

požadavky daly snadno dohledat. Dohledatelnost se nejlépe vyřeší tak, že všechny případy užití a funkční požadavky uložíte do nějakého specializovaného nástroje.

Další informace

O nástrojích pro správu požadavků se více dozvíte v 21. kapitole.

Pouze specifikace

Třetí možností je uspořádat specifikaci podle funkcí systému nebo případů užití a zahrnout do ní kromě funkčních požadavků i případy užití. (Pro toto řešení se rozhodl tým Systému pro evidenci chemikálií.) Samostatné případy užití se při tomto přístupu vůbec nepoužívají. Případné duplicitní požadavky musíte jasně označit a nebo každý požadavek důsledně uvádět jen jednou, a kdykoliv se ocitne v dalším případu užití, odkázat se na něj.

Výhody případů užití

Z praxe

Hlavní síla případů užití je v tom, že se soustředí na uživatele a jejich úkoly. Když při zjišťování požadavků na nový systém začnete uživateli namísto funkcemi, uživatelé budou lépe vědět, co mohou od systému očekávat. Zúčastnil jsem se několika internetových projektů a zástupci jejich uživatelů se shodli, že si díky případům užití ujasnili, co by vlastně měl jejich web návštěvníkům nabízet. Případy užití pomáhají analytikům a vývojářům pochopit aplikační doménu a uživatelovo podnikání. Psaním testovacích scénářů podle případů užití a pečlivým promyšlením dialogu mezi aktéry a systémem můžete už na začátku vývoje najít nejednoznačné požadavky.

Psaní kódu, který se nikdy nepoužije, je zbytečné a demotivuje vývojáře. Když požadavky předem naddimenzujete a budete se do nich snažit zahrnout každou myslitelnou funkci, některé z požadavků se nejspíš budou implementovat zbytečně. Když ale vyjdete od případů užití, dostanete jen funkční požadavky, které uživatelé nezbytně potřebují. Vyhýbáte se tím opuštěným funkcím, které během sbírání požadavků vypadaly slibně, ale jež nikdo nepoužívá, protože nejsou potřeba pro žádný z uživatelských úkolů.

Případy užití také pomáhají s rozdělením důležitosti požadavků. Nejdůležitější funkční požadavky jsou ty, které pocházejí z nejdůležitějších případů užití. Případ užití může být důležitý z několika důvodů:

- Popisuje některý ze základních podnikatelských procesů, o něž se má systém postarat.
- Používá jej hodně uživatelů a často.
- Řekla si o něj přednostní třída uživatelů.
- Je důležitý pro splnění nějakého předpisu.
- Spoléhají na něj jiné systémy.

Upozornění

Neztrácejte čas pilováním případů, které se budou implementovat až za několik měsíců nebo let. Jejich zadání se do té doby stejně nejspíš změní.

Případy užití mají i čistě technické výhody. Z jejich perspektivy jsou dobře vidět některé důležité doménové objekty a jejich vzájemné vztahy. Vývojářům, kteří používají objektový návrh, se případy užití dobře převádí na objektové modely, například diagramy tříd a sekvenční diagramy. (Samozřejmě ale platí, že případy užití nejsou v žádném případě omezené na projekty vyvíjené objektově.) Podnikatelské procesy se časem mění, takže se budou měnit i úkoly popsané v různých případech užití. Pokud se všechny vaše funkční požadavky, návrhy, kód i testy dají dohledat až k původním případům užití (tedy hlasu zákazníka), budou se vám tyto změny snáze šířit celým systémem.

Na co si dát pozor

Případy užití se – stejně jako kterýkoliv z ostatních nástrojů softwarového inženýrství – dají použít špatně (Kulak a Guiney 2000, Lilly 2000). Dávejte si pozor na následující pasti:

- **Příliš mnoho případů užití.** Uprostřed stovek případů užití člověk snadno sklouzne k nevhodné úrovni abstrakce. Nepište nový případ užití pro každý scénář. Z běžné cesty, alternativních cest a všech výjimek udělejte jen scénáře jednoho společného případu užití. Případů užití většinou bývá mnohem víc než podnikatelských požadavků a funkcí, ale mnohem méně než funkčních požadavků.

Z praxe

Příliš složité scénáře. Jednou jsem dostal do ruky případ užití, který zabíral čtyři hustě popsané stránky s popisem dialogů, rozhodovací logiky a podmínek větvení. Byl naprosto nesrozumitelný. Složitost podnikatelských úkolů neovlivníte, ale způsob jejich zápisu ano. Vyberte si jednu úspěšnou cestu skrz případ užití, jediné dosazení do všech logických proměnných, a označte ji jako běžnou cestu. Ostatní úspěšné větve budou alternativní cesty, neúspěšné větve výjimky. Alternativních cest může být hodně, ale každá z nich musí být krátká a srozumitelná. Abyste popis případu zbytečně nekomplikovali, zapisujte jen to nejdůležitější chování aktérů a systému; nepodstatné podrobnosti vynechte.

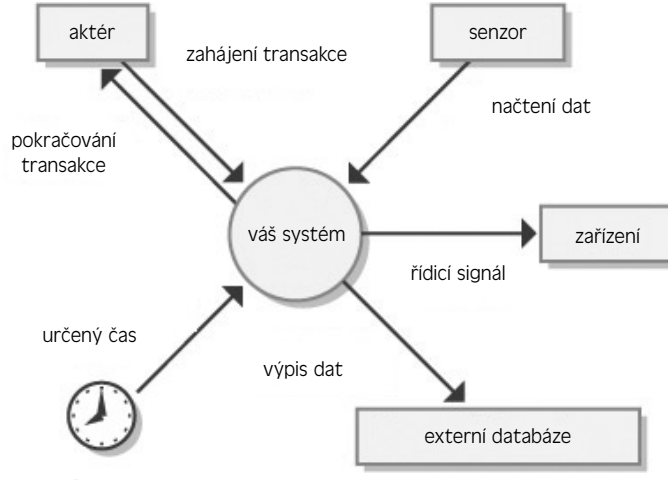
- **Návrhy uživatelského rozhraní v případech užití.** Případy užití by se měly soustředit na potřeby uživatele, nikoliv na vzhled dialogů. Zdůrazňujte *principy* komunikace mezi aktéry a systémem, a řešení konkrétních částí rozhraní si nechte až do fáze návrhu. Místo věty „systém zobrazí rozbalovací seznam“ můžete použít například „systém zobrazí seznam možností“. Nedopusťte, aby se analýza požadavků přizpůsobovala návrhu rozhraní. Náčrtky obrazovek a mapy dialogů (neboli diagramy uživatelského rozhraní, viz kapitolu 11) by měly sloužit jen pro dokreslení interakce mezi aktéry a systémem, nikoliv jako závazná specifikace.

- **Definice dat v případech užití.** Viděl jsem případy užití, které obsahovaly definici všech potřebných datových struktur. Při takovém přístupu se definice datových struktur špatně hledají, protože není jasné, který případ užití hledanou definici obsahuje. Navíc zde dochází ke zdvojování jednotlivých definic, které se pak při změnách mohou snadno rozejít. Definice dat proto nerozptylujte mezi případy užití a raději je všechny soustřeďte do projektového datového slovníku (viz kapitolu deset).
- **Případy užití, kterým uživatelé nerozumí.** Pokud si uživatel nedokáže dát případ užití dohromady se svými úkoly nebo podnikatelskými procesy, chyba je v případu užití. Případy užití pište z pohledu uživatele, nikoliv systému, a průběžně si je nechávejte uživateli kontrolovat. Pokud jde o plně formálně popsaný případ užití, možná je pro váš projekt zbytečně složitý?
- **Nové podnikatelské procesy.** Pokud má software podporovat nějaké podnikatelské procesy, které ještě neexistují, uživatelé budou mít s popisováním případů užití potíže. V takových případech není sbírání požadavků ani tak modelování podnikatelských procesů, jako spíš jejich vymýšlení. Nechávat zavádění efektivních podnikatelských procesů na novém informačním systému je velké riziko, takže byste měli nejdříve dát dohromady podnikatelské procesy, a teprve pak plně specifikovat systém.
- **Příliš časté vkládání a rozšiřování případů užití.** Při prvním setkání s případy užití musí analytici, uživatelé i vývojáři změnit svůj způsob uvažování o požadavcích. Některé nuance vkládání a rozšiřování případů užití jsou v takových případech zbytečně složité. Nejlépe uděláte, když se jim vyhnete a počkáte, dokud se s případy užití nebudete cítit dostatečně pohodlně.

Tabulky událostí a odpovědí

Uživatelské požadavky se dají uspořádat a dokumentovat také nalezením všech vnějších událostí, na které musí systém odpovědět. *Událost* je změna nebo činnost, která probíhá v uživatelském prostředí a žádá si od systému nějakou odpověď (McMenamin a Palmer 1984, Wiley 2000). *Tabulka událostí a odpovědí* (nebo též *tabulka událostí* nebo *seznam událostí*) obsahuje všechny události a chování, které se od systému při dané události očekávají. Existuje několik typů systémových událostí, viz obrázek 8.9:

- Uživatelské akce, které vedou k nějakému dialogu se systémem (například když uživatel zahájí případ užití). Někdy se označují jako *podnikatelské události*. Tabulka událostí u nich odpovídá popisu dialogu v případu užití. Na rozdíl od případů užití ale nepopisuje uživatelův cíl ani hodnotu, kterou mu posloupnost událostí a odpovědí přinese.
- Řídicí signály, načítání dat nebo přerušení z vnějšího hardwarového zařízení, například sepnutí přepínače, změna napětí nebo pohyb myši.
- Časová událost, například dosažení určitého času (automatický půlnoční export dat a podobně) nebo uplynutí nějaké lhůty od poslední události (například u systému, který každých deset sekund zapisuje senzorem naměřenou teplotu).



Obrázek 8.9. Příklady systémových událostí a odpovědí.

Tabulky událostí jsou zvlášť vhodné pro real-time systémy. Příkladem tabulky událostí, která částečně popisuje chování stěračů, je tabulka 8.1. Všimněte si, že očekávaná odpověď závisí nejen na události, ale i na stavu systému v okamžiku jejího příchodu. Například odpověď na události 4 a 5.1 záleží na tom, jestli už byly stěrače při zapnutí cyklování zapnuté. Odpovědí může být jednoduše změna nějakých vnitřních systémových informací (události 4 a 7.1) a nebo nějaká viditelná odpověď (většina ostatních událostí).

Tabulka 8.1 Systém pro řízení stěračů, tabulka událostí.

ID	Událost	Stav systému	Odpověď systému
1.1	nastavit pomalý chod stěračů	stěrače vypnuté	nastavit motor stěračů na pomalý chod
1.2	nastavit pomalý chod stěračů	stěrače v rychlém chodu	nastavit motor stěračů na pomalý chod
1.3	nastavit pomalý chod stěračů	stěrače cyklují	nastavit motor stěračů na pomalý chod
2.1	nastavit rychlý chod stěračů	stěrače vypnuté	nastavit motor stěračů na rychlý chod
2.2	nastavit rychlý chod stěračů	stěrače v pomalém chodu	nastavit motor stěračů na rychlý chod
2.3	nastavit rychlý chod stěračů	stěrače cyklují	nastavit motor stěračů na rychlý chod
3.1	vypnout stěrače	stěrače v rychlém chodu	dokončit stírací cyklus, vypnout motor
3.2	vypnout stěrače	stěrače v pomalém chodu	dokončit stírací cyklus, vypnout motor
3.3	vypnout stěrače	stěrače cyklují	dokončit stírací cyklus, vypnout motor

ID	Událost	Stav systému	Odpověď systému
4	nastavit stěrače na cyklování	stěrače vypnuté	načíst časový interval, inicializovat časovač
5.1	nastavit stěrače na cyklování	stěrače v rychlém chodu	načíst časový interval, dokončit stírací cyklus, inicializovat časovač
5.2	nastavit stěrače na cyklování	stěrače v pomalém chodu	načíst časový interval, dokončit stírací cyklus, inicializovat časovač
6	uběhl stanovený interval od posledního stíracího cyklu	stěrače cyklují	provést jeden pomalý stírací cyklus
7.1	změna cyklovacího intervalu	stěrače cyklují	načíst časový interval, inicializovat časovač
7.2	změna cyklovacího intervalu	stěrače vypnuté	bez odezvy
7.3	změna cyklovacího intervalu	stěrače v rychlém chodu	bez odezvy
7.4	změna cyklovacího intervalu	stěrače v pomalém chodu	bez odezvy
8	signál pro okamžité setření	stěrače vypnuté	provést jeden pomalý stírací cyklus

Tabulka událostí zachycuje informace na úrovni uživatelských požadavků. Pokud obsahuje všechny možné kombinace událostí, stavů a odpovědí systému (včetně výjimečných stavů), může posloužit i jako doplněk funkčních požadavků na danou část systému. Analytik ovšem musí ve specifikaci v každém případě uvést další funkční i parametrické požadavky. Kolik cyklů například stěrače zvládnou za jednu minutu v pomalém a rychlém režimu? Dá se interval cyklování nastavit plynule, nebo je odstupňovaný v nějakých krocích? Jaká je nejkratší a nejdelší povolená doba mezi dvěma cykly? Kdybyste specifikaci požadavků skončili uživatelskými požadavky, vývojáři by si podobné informace museli shánět sami. Pamatujte, vaším cílem je popsat požadavky tak přesně, aby vývojáři věděli, co mají napsat!

Všimněte si, že události z tabulky 8.1 popisují princip události. Jsou popsány na *principiální* úrovni, a nikoliv na úrovni *implementační*, kde by popisovaly konkrétní podrobnosti jejich implementace. Tabulku 8.1 nezajímá, jak vypadají samotné ovládací prvky, ani jak s nimi uživatel pracuje. Designér by tyto požadavky mohl převést na cokoliv od klasických páček na sloupku volantu až po hlasové rozpoznávání příkazů *zapnout stěrače*, *vypnout stěrače*, *stěrače rychle*, *setřít sklo* a podobně. Psaním požadavků na principiální úrovni se vyhýbáte zbytečnému omezování návrhu. Pokud ale nějaké objektivní omezení návrhu skutečně existuje, určitě na něj designéra v dokumentaci upozorněte.

Další kroky

- Podle šablony z obrázku 8.6 napište několik případů užití pro svůj aktuální projekt. Nezapomeňte najít všechny alternativní cesty a výjimky. Vypracujte funkční požadavky, které uživatelům umožní všechny tyto případy užití úspěšně dokončit. Podívejte se, jestli vaše aktuální specifikace tyto funkční požadavky obsahuje.
- Napište seznam vnějších událostí, na které váš systém musí odpovídat. Sestavte tabulku událostí, která bude popisovat všechny stavy systému během přijetí události a způsob odpovědi.