OBJEKTOVO-ORIENTOVANÁ ANALÝZA/NÁVRH SYSTÉMOV A MODELOVANIE DÁTOVÝCH TOKOV*

(OBJECT-ORIENTED ANALYSIS/DESIGN OF SYSTEMS AND DATA FLOW MODELING)

Zdeněk HAVLICE, Ján KOLLÁR, Vladimír CHLADNÝ
Katedra počítačov a informatiky, Fakulta elektrotechniky a informatiky Technickej univerzity v Košiciach, Letná 9, 042 00 Košice, Slovak Republic, tel. 055/602 2568, E-mail: zdenek.havlice@tuke.sk, jan.kollar@tuke.sk, vlado.chladny@tuke.sk

SUMMARY

Structured, modular and object-oriented paradigms that are applied for software systems analysis, design and implementation, represent subsidiary principles. This article describes the close relations between structured and object-oriented approaches. It also presents the possibilities and advantages of a structured view to a software system using the Dataflow Diagrams (DFD), when Object-Oriented Analysis and Design (OOA/D) are performed. Extension of the Unified Modeling Language (UML) by the DFD is presented on simple examples. Relationships and dependencies between the models when system architecture, system linkage to surround environment, system services and system Graphical User Interface (GUI) are modeled are presented as well.

Keywords: *object-oriented analysis/design, OOA/D, UML, dataflow modeling, dataflow diagram, DFD, CDFD, CASE*

1. ÚVOD

Rastúci výkon technických prostriedkov výpočtových systémov vytvára jeden z predpokladov pre implementáciu stále rozsiahlejších a zložitejších systémov. Druhým predpokladom pre ich úspešnú implementáciu je zvládnutie práve tejto zložitosti a rozsahu. Preto sa venuje veľká pozornosť výskumu a vývoju v oblasti metód, nástrojov a metodológií aplikovateľných v celom životnom cykle softvérových systémov. Existujú staršie klasické postupy, ktoré už ukázali svoje výhody a sú dostatočné známe ich nevýhody a riziká spojené s ich použitím. Práve preto môže byť zaujímavé a užitočné využitie vhodnej kombinácie moderných a klasických metód.

Príspevok ukazuje výhody využitia kombinácie objektových a štruktúrovaných modelov pri analýze a návrhu programových systémov.

1.1 Objektový a štruktúrovaný pohľad na systémy

Reálny aj imaginárny **svet systémov tvoria objekty** s charakteristickými vlastnosťami a správaním. V tejto integrálnej podobe sú známe svojim používateľom a takto im používatelia aj najlepšie rozumejú. Analytická podoba systémov a ich častí vo forme rôznych abstraktných analytických modelov je spravidla používateľom veľmi vzdialená.

Vzhľadom na nevyhnutnosť priebežnej konfrontácie analytických a návrhárskych prác s požiadavkami používateľov je nesporne užitočné využiť pri analýze/návrhu systémov objektovo - orientovaný prístup (OOA/D). **Úžitok OOA/D je predovšetkým vo**:

- **vyššej zrozumiteľnosti** pre budúcich používateľov systémov,
- **minimalizácii divergencie modelov** reprezentujúcich rôzne pohľady na systém,
- **priamočiarom prechode k objektovej implementácii .**

Implementačné vývojové systémy (integrované vývojové prostredia - IDE, systémy pre rýchly vývoj aplikácií – RAD), napr. [1,2,3] sú založené na objektových princípoch. Umožňujú tak využiť prínos objektového prístupu pri implementácii pre:

- **vyššiu znovupoužiteľnosť** artefaktov vývoja (využitie dedičnosti, knižníc tried, šablón a komponentov),
- **vývoj bezpečnejších a spoľahlivejších aplikácií** (zapuzdrenie, zakrývanie implementácie, interfejsy, ošetrenie výnimiek, automatické riadenie dynamickej pamäti).

OOA/D umožňuje zabezpečiť priamočiary prechod od analytických a návrhových modelov zostavených na báze OO paradigmy k objektovo orientovanej implementácii (OOI). Uplatnenie rovnakej paradigmy pri analýze, návrhu aj implementácii je výhodou, ale nie nutnosťou. Umožňuje využiť pozitíva OO prístupu v analýze, návrhu aj implementácii kontinuálne. Štruktúrovaná analýza a návrh (SA/D) je v súčasnosti podporovaná už len staršími verziami CASE systémov. Naproti tomu štruktúrovaná implementácia nie je viazaná len k prežívajúcim jazykom štruktúrovaného programovania, ale aj k objektovo - orientovaným jazykom. **Štrukturálne prvky** ako sú údajové štruktúry, riadiace štruktúry, dátové a riadiace toky, dátové pamäte a procesy sú štrukturálnymi prvkami aj v objektovo implementovaných systémoch. **Väzba objektov na štrukturálne prvky vyplýva zo spôsobu definovania tried a objektov a zo spôsobu ich použitia v systémoch**:

* *Výskum s podporou grantového projektu VEGA 1/9026/02*

- Členské dáta tried sú definované ako **údajové štruktúry**, členské funkcie používajú **údajové a riadiace štruktúry**.
- Objektovo implementovaný programový systém tvoria objekty, ktoré medzi sebou komunikujú prostredníctvom udalostí a správ. Obsahom udalostí aj správ sú **údajové štruktúry**.
- Objekty využívajú pre vzájomnú komunikáciu **dátové a riadiace toky**. Tieto toky prenášajú informácie vo forme údajových štruktúr a vznikajú na základe definovaných vlastností a správania objektov.
- Vykonávaním členských funkcií a komunikáciou medzi objektmi sa realizujú **procesy**, ktoré sú zaujímavé z pohľadu používateľa (podnikové, administratívne, výrobné, riadiace, ...) ale aj z hľadiska implementácie systému (procesy na úrovni operačného systému, ich vzájomná komunikácia).
- Objekty uchovávajú svoje vlastnosti a aktuálny stav rezidentne v rôznych **dátových pamätiach** (súboroch, relačných a objektových databázach).

Štrukturálne prvky SA/D: **procesy**, **dátové a riadiace toky** môžu byť použité ako nástroje pre objektové modelovanie na vyššej úrovni abstrakcie.

Štrukturálne prvky SA/D: **riadiace štruktúry** (algoritmy) a **údajové štruktúry** môžu byť použité ako nástroje pre objektové modelovanie na nižšej úrovni abstrakcie.

1.2 Dátové toky, riadiace toky, procesy a objekty

Jednotlivé **procesy** sa realizujú interakciou objektov na báze posielania/prijímania správ, generovania udalostí a reakcií na ne. To je detail zaujímavý buď pri detailnej analýze alebo až pri implementácii návrhových modelov. Elementárne správy a udalosti bývajú obyčajne súčasťou rozsiahlejších štruktúr, ktoré tvoria podstatu komunikácie vo forme štruktúrovaných dátových a riadiacich tokov. **Dátové a riadiace toky** sú podstatou všetkých procesov – sú ich vstupom a/alebo výstupom a predmetom spracovania. Preto sa zdá prirodzené, aby medzi návrhovými nástrojmi pre modelovanie programových systémov boli prostriedky pre modelovanie týchto tokov na vyšších úrovniach abstrakcie ako je elementárna úroveň udalostí alebo správ medzi objektmi.

2. ROZŠÍRENIE UML O PRVKY SA/D

Napriek úzkemu vzťahu objektov, procesov, dátových a riadiacich tokov v súčasnosti uznávaný štandard pre objektové modelovanie - UML [4,5] zahŕňa žiadne z tradičných nástrojov štrukturálnej analýzy a návrhu (hoci niektoré zdrojové staršie OO metódy tieto nástroje využívali , napr. Shlaer/Mellor Method [6]).

Modelovanie dátových tokov je možné pomocou nástrojov UML na elementárnej úrovni ako:

- časová následnosť posielaných správ v **interakčnom diagrame sekvencií** alebo **diagrame spolupráce**,
- obsah správ a udalostí vo forme popisu **parametrov volaných členských funkcií tried**.

To, čo je možné popísať jazykom UML v diagrame sekvencií alebo v diagrame spolupráce, je na vysokej úrovni detailizácie – **časová následnosť posielaných správ**. Na vyššej úrovni abstrakcie obyčajne ale postačuje informácia o tom, aké vstupy a výstupy analyzovaný proces má, akú majú štruktúru a domény pre jednotlivé zložky. To nie je možné jednoducho popísať ani modelom na báze UML diagramu prípadov použitia (USCD -Use Case Diagram), ani interakčnými diagramami UML (SEQD-Sequence Diagram, COLD-Collaboration Diagram).

Existuje taký štruktúrovaný pohľad na systém, ktorý neznehodnotí vytvorené objektové modely v žiadnej z oblastí, pre ktoré je OO prístup považovaný všeobecne za užitočný: nie je rizikom ich divergencie, menšej zrozumiteľnosti a nemá ani negatívny vplyv na objektovú implementáciu. Súčasne je prínosom pre vytvorenie komplexnejšieho systému abstraktných pohľadov na analyzovaný/navrhovaný systém. Jedná sa o **tradičný štruktúrovaný pohľad na báze de Markovho modelu dátových tokov** (DFD – Data Flow Diagram), resp. jeho rozšírení (CDF – Control Data Flow Diagram). Objektovosť prístupu nebude narušená, ak v analýze a návrhu budú *procesy viazané k objektom*, ktoré ich vytvárajú. Proces v DFD je v tomto smere iba vhodným prostriedkom na zgrupovanie úzko súvisiacich objektov. Spoluprácou objektov sa realizuje procesom definovaná činnosť (prípad použitia, služba, aktivita). Pritom je k dispozícii možnosť hierarchickej dekompozície od globálnych procesov na najvyššej úrovni abstrakcie až po elementárne procesy.

Reálny svet tvoria objekty - to je fakt, ktorý sa dá zmeniť v predstavách analytikov, ak je to potrebné, pri analýze a návrhu systémov. Objekty môžu byť rozložené na potrebný počet analytických pohľadov s účinnou podporou CASE systémov schopných sledovať vzájomné súvislosti a konzistenciu modelov navzájom.

Metodológie OOA/D postavené na UML sa vyhýbajú tradičnému nástroju štruktúrovaného modelovania procesov na báze DFD, hoci tento model sám o sebe:

- je dostatočne zrozumiteľný (s definovateľnou úrovňou detailizácie prostredníctvom hierarchickej dekompozície procesov),
- dostatočne presný pre popis rozhraní procesov (vstupy/výstupy) a smerov komunikácie (zdroje a ciele dátových tokov),

- nespôsobuje riziká divergencie dátového a procesného pohľadu (procesy predstavujú interakciu objektov a dáta v dátových pamätiach a dátových tokoch sú viazané na objekty, resp. triedy, ktoré dátové aj funkčné vlastnosti objektov deklaratívne integrujú v sebe).

Naviac „bublinový diagram“ (buble chart) - klasická reprezentácia DFD je intuitívna forma komunikácie medzi odborníkmi aj medzi laikmi. K takejto forme modelu dospeje v podstate takmer každý pokus o vysvetlenie toho, ako existujúci systém funguje alebo ako nový systém má fungovať.

Ak proces definujeme ako:

1. vonkajší prejav existencie objektov, ktoré navzájom komunikujú – **interaktívny proces** v DFD (zabezpečujúci vstupy/výstupy dát)
2. vnútorné dôsledky existencie objektov, ktoré navzájom komunikujú – **neinteraktívne procesy** v DFD (procesy aplikačnej vrstvy systému, ktoré sú používateľovi zakryté)

potom model na báze CDFD (Control Data Flow Diagram) je vhodným prirodzeným rozšírením UML modelov. Doplňuje pôvodný systém objektových modelov, ktoré sa týkajú objektovo chápaného systému nasledovne:

- **modelovanie podnikových, riadiacich a iných procesov** analyzovaného resp. navrhovaného systému na konceptuálnej úrovni,
- **modelovanie služieb systému** - spojenie jednotlivých súvisiacich prípadov použitia systému do interaktívnych a neinteraktívnych procesov s presne definovanými vstupmi a výstupmi.

2.1 Modelovane architektúry systému a väzieb na okolie

Súčasné informačné a riadiace systémy sú najčastejšie integrované z rôznych podsystémov vyvíjaných a implementovaných často použitím rôznych technológií.

Pre návrh **architektúry integrovaných informačných a riadiacich systémov** (IaRS) je kritický návrh a implementácia **zdieľaných dátových pamätí**, **komunikácie** vo vnútri systému a komunikácie systému s okolím. Pre tento účel je dôležité vymedzenie nasledujúcich množín modelov:

- model podsystémov (SubS) a ich vzájomných väzieb,
- model zdieľaných dátových pamätí (ShDS),
- model okolia systému – externých entít (EE), ktoré tvoria: externé systémy (ExtS) a používatelia (Usrs),
- model komunikácie medzi podsystémami, okolím a dátovými pamäťami – riadiace a dátové toky (CtrF, DatF).

Konceptuálny Model architektúry (CAM) pre IaRS je na základe toho možné definovať ako usporiadanú šesticu množín modelov:

CAM = (SubS, ShDS, ExtS, Usrs, CtrF, DatF) ,
kde:

$$\mathbf{SubS} \neq \varnothing$$
$$\mathbf{Usrs} \cup \mathbf{ExtS} \neq \varnothing$$
$$\mathbf{DatF} \cup \mathbf{CtrF} \neq \varnothing$$

- t.j. existuje aspoň jeden model subsystému (ak je práve jeden, predstavuje model celého systému) a aspoň jeden model prvku okolia, s ktorým systém komunikuje aspoň jedným riadiacim alebo dátovým tokom.

Pre zostavenie CAM je možné použiť DFD. Takto zostavený štrukturálny model architektúry systému (S-CAM) poskytuje aj možnosti modelovať:

- **konceptuálnu architektúru** budúceho alebo existujúceho systému bez ohľadu na implementačné detaily,
- **kritické aspekty** distribuovaných a/alebo heterogénnych systémov (komunikácia medzi podsystémami, zdieľané dátové pamäte, komunikácia s okolím),
- **prepojenie konceptuálnych prvkov architektúry s implementačnými detailami** (funkcie, procedúry, údajové štruktúry, objekty, komponenty, servre).

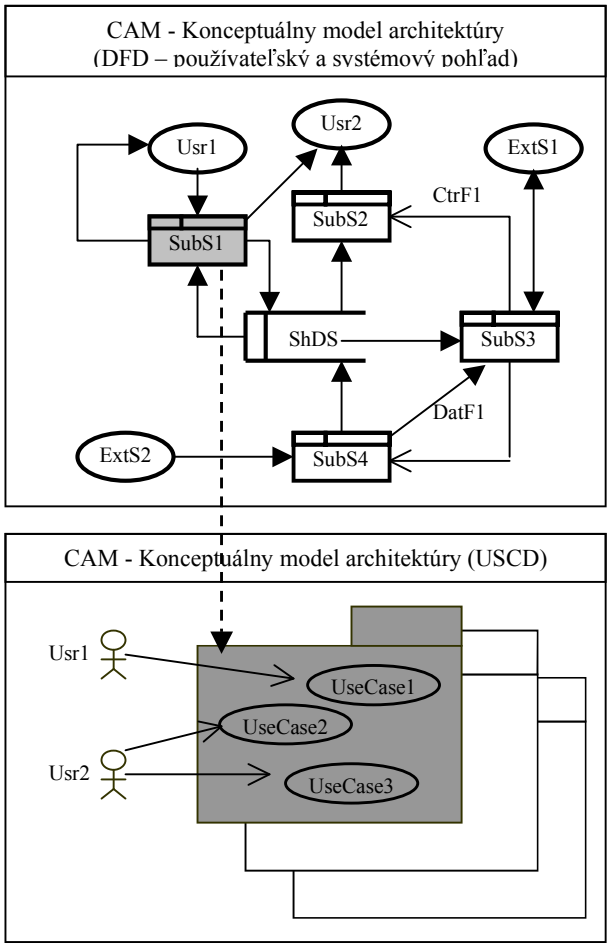
Ak sa pre zostavenie CAM použijú objektové modely z UML, zostavený **objektový model architektúry** (OO-CAM) umožňuje modelovať:

- **role prvkov okolia pri komunikácii so systémom** – iniciátori a/alebo používatelia služieb systému – use-case diagram (USCD),
- **štruktúru dátových pamätí** (objektovú, relačnú, hierarchickú) – pomocou diagramu tried (CLAD) s perzistentnými triedami a reláciami medzi nimi.

MODEL architek-túry	Modelovací nástroj SA/D (modelovací prvok)	Modelovací nástroj OOA/D (modelovací prvok)
Subs	DFD (proces) spoločný pohľad na podsystémy a ich vzájomné väzby s možnosťou dekompozície možný používateľský aj návrhársky -systémový pohľad	USCD (prípad použitia podsystému alebo balík takýchto prípadov) len používateľský pohľad
ShDS	DFD (dátová pamäť) spoločný pohľad na podsystémy , zdieľané dáta a ich toky	CLAD (trieda) pohľad na zdieľané dáta aj s povolenými operáciam
ExtS	DFD (externá entita)	USCD (aktér)
Usrs	DFD (externá entita)	USCD (aktér)
CtrF	CDFD (riadiaci tok)	SEQD, COLD
DatF	DFD (dátový tok)	SEQD (argumenty správ)

Tab. 1 Štruktúrované a objektové prvky modelu architektúry
Tab. 1 Structural and object elements of architecture model

V Tab. 1 je prehľad modelovacích prvkov pre modelovanie architektúry použitím UML rozšíreného o DFD. Súvislosť modelov je na Obr. 1.



Obr. 1 Štruktúrované a objektové modely architektúry
Fig. 1 Structured and object architecture models

2.2 Modelovane služieb systému

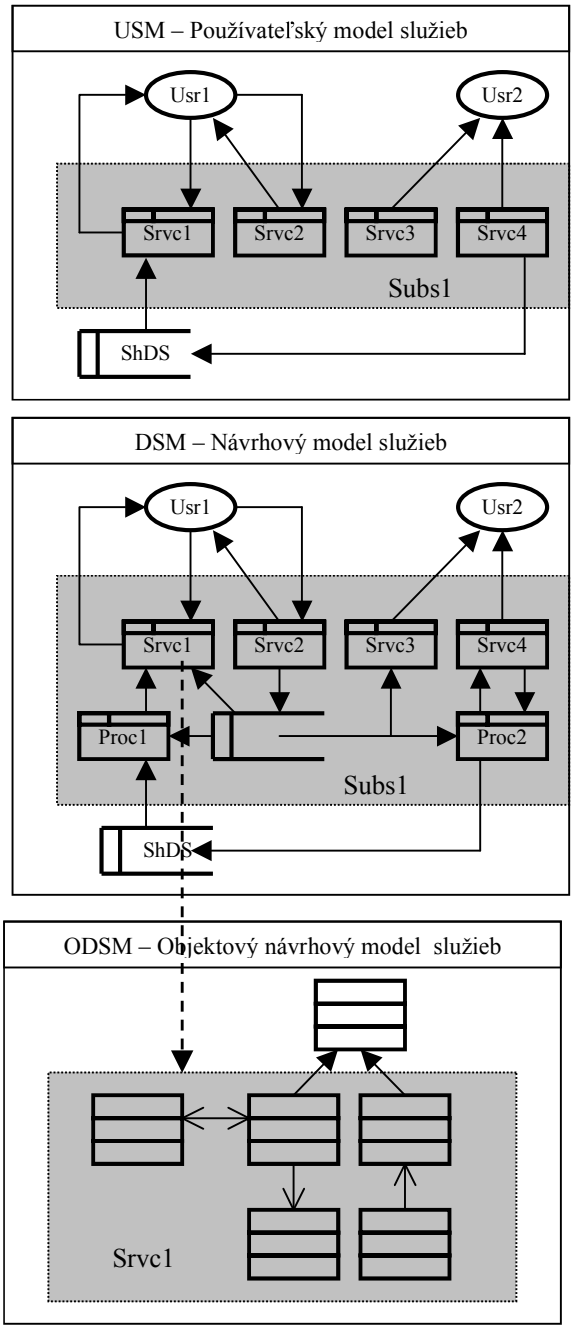
USCD (Use Case Diagram) – sa používa pre zostavenie modelu prípadov použitia systému. Ak majú prípady charakter služieb, ktoré systém poskytuje používateľovi, je možné tento model považovať za model ponuky služieb systému. Vzhľadom na svoju jednoduchosť a prehľadnosť je ako semiformálny model určený predovšetkým na konfrontáciu používateľských požiadaviek s návrhom ich zakomponovania do návrhu výsledného riešenia.

Model služieb systému zostavený pomocou USCD pre desiatky a stovky služieb reálnych rozsiahlych systémov stráca svoje opodstatnenie v grafickej podobe, pokiaľ sa nepoužije mechanizmus na ich zgrupovanie pomocou balíkov.

Mechanizmus hierarchickej dekompozície procesov DFD dovoľuje modelovať všetko to, čo aj USCD (Tab. 2). Naviac DFD môže dobre modelovať služby nielen z pohľadu používateľa, ale aj z analytického pohľadu štruktúry systému

MODEL služieb	Modelovací nástroj SA/D (modelovací prvok)	Modelovací nástroj OOA/D (modelovací prvok)
Srvc - služba z používateľského pohľadu	DFD (proces)	USCD (prípad použitia)
Producent dát	DFD (externá entita)	USCD (aktér)
Iniciátor služby – aktivácia	DFD (externá entita)	USCD (aktér)
Konzument dát	DFD (externá entita)	USCD (aktér)
Realizácia služby (systémový-návrhársky pohľad)	DFD (proces)	CLAD (triedy a objekty), SEQD, COLD (interakcia objektov)

Tab. 2 Štruktúrované a objektové prvky modelu služieb
Tab. 2 Structural and object elements of services model



Obr. 2 Štruktúrované a objektové modely služieb
Fig. 2 Structural and object services models

a informačných tokov. Ak sa v UML použije na doplnenie informácií o službách popísaných v USCD aj diagram dátových tokov (DFD), môže byť tento model prepojovacím mostom medzi používateľským a systémovým (analytickým, návrhovým, implementačným) pohľadom na služby systému.

Používateľský model služieb (USM) na báze DFD obsahuje len procesy odpovedajúce skupinám služieb pre jednotlivé kategórie používateľov. Návrhový model realizácie služieb pomocou procesov (DSM) je rozšírením USM o vnútorné procesy, toky a dátové pamäte. Objektový návrhový model služieb (ODSM) zostavený použitím diagramu tried (CLAD) a interakčných diagramov SEQD, COLD je rozšírením štrukturálneho pohľadu na služby o pohľad objektový. Popisuje objekty (resp. triedy objektov), aktivitou ktorých je služba poskytovaná alebo realizovaná.

V Tab. 2 je prehľad modelovacích prvkov pre modelovanie služieb použitím UML rozšíreného o DFD. Súvislosť modelov je na Obr. 2.

2.3 Modelovanie používateľského rozhrania

Model používateľského rozhrania (UIM) je rozšírením používateľského modelu služieb (USM). Mal by umožniť popis 3 kritických pohľadov na rozhranie:riadenie interakcie , vizuálna reprezentácia – vzhľad, štruktúra a obsah (syntax a sémantika) vstupov a výstupov .

STCD (Statechart Diagram) modeluje prechody medzi stavmi dialógu interaktívnej služby. Každý stav odpovedá jednému komunikačnému oknu s interakčnými objektmi pre vstup/výstup údajov a riadenie interakcie.

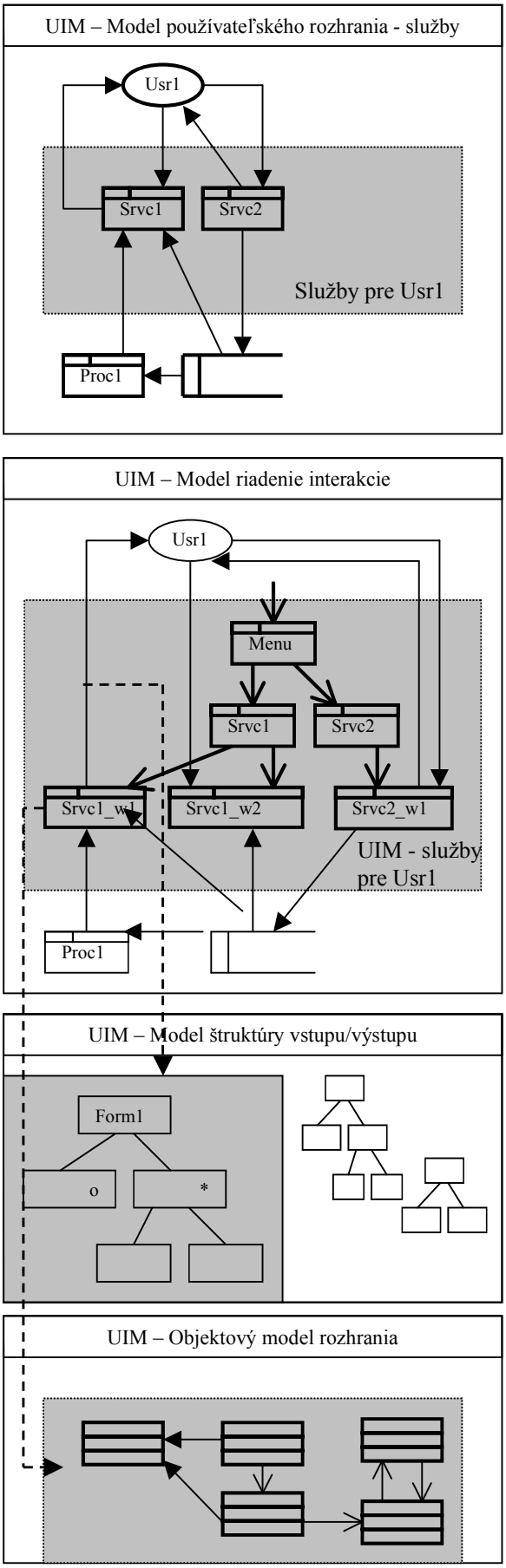
CDFD (Control Dataflow Diagram) modeluje interaktívne služby z hľadiska dátových vstupov, výstupov, zdrojov a cieľov dátových prenosov.

DSD (Data Structure Diagram) modeluje štruktúru vstupu/výstupu.

CLAD (Class Diagram), OBJD (Object Diagram) – model stavu dialógu – modeluje vizuálne interakčné objekty a nevizuálne aplikačné objekty v komunikačnom okne.

MODEL používateľského rozhrania	Modelovací nástroj SA/D (modelovací prvok)	Modelovací nástroj OOA/D (modelovací prvok)
Interakčný (dialógový) prvok (tlačidlo, zoznamové pole,. vstupné pole..)		CLAD (trieda, objekt)
Stav dialogu (komunikačné okno, frame, príkazový riadok,...)	STCD (stav) alebo DFD (proces)	STCD (stav) CLAD (väzby medzi objektami rozhrania
Riadenie interakcie	STCD (prechody medzi stavmi) alebo CDFD (riadiace toky)	STCD (prechody medzi stavmi) SEQD, COLD (komunikácia objektov rozhrania)
Štruktúra vstupov/výstupov	DSD	CLAD (hierarchia tried a atribúty tried)

Tab. 3 Štruktúrované a objektové prvky modelu používateľského rozhrania
Tab. 3 Structural and object elements of user interface model



Obr. 3 Štruktúrované a objektové modely rozhrania
Fig. 3 Structural and object interface models

Interakčné diagramy UML - SEQD a COLD umožňujú detailne popísať interakciu objektov, ktoré sa na vytváraní používateľského rozhrania zúčastňujú.

Ak je zostavený STCD pre každého aktéra z USCD - používateľa s interaktívnym prístupom k službám systému a naviac každý stav v STCD má definovaný súvisiaci štrukturálny model stavu dialógu pomocou CLAD, potom môže byť súvisiaca trojica modelov USCD-STCD-CLAD použitá ako formálny model pre generovanie zdrojových textov metód pre komunikačné okno aplikácie s danou ponukou služieb. Takýto postup sa ale nepoužíva z praktických dôvodov – efektívnejšia je syntéza zdrojových textov používateľského rozhrania aplikácie použitím WYSIWYG prístupu v RAD systémoch. Jednoduchým vytváraním komunikačných okien a priamym umiestňovaním interakčných objektov v nich.

3. VÝZNAM KONFIGUROVATEĽNOSTI CASE SYTÉMOV

Medzi účinné nástroje predchádzajúce problémom so spoľahlivosťou a inými vonkajšími vlastnosťami programových systémov patria modely a experimentovanie s nimi. Reálne systémy mávajú veľa rôznych vlastností, niektoré sú podstatné, iné menej. Rôznorodosť týchto vlastností závisí od konkrétnych systémov, podmienok, v ktorých majú systémy fungovať, od ich zložitosti a iných činiteľov. V súčasnej dobe sa rýchlo rozvíjajú a široko uplatňujú distribuované systémy. Jednou prudko sa rozvíjajúcou oblasťou sú napríklad e-learning systémy, ktoré transformujú výučbový proces [9]. Táto rôznorodosť ovplyvňuje množstvo abstraktných pohľadov na systém, ktoré je potrebné vytvoriť, preto aby mohli byť systémy z týchto pohľadov popísané, namodelované, špecifikované a aby boli odsimulované všetky tie vlastnosti budúceho systému, ktoré budú kritické pre jeho úspešnosť. Rôzne pohľady na systém vyžadujú obyčajne aj rôzne modelovacie nástroje. Nie je možné, ani by nebolo účelné hľadať univerzálnu konečnú nemennú sadu postačujúcich druhov modelov. Preto aj definované modelovacie a špecifikačné jazyky sa neustále rozširujú, modifikujú a doplňujú. Investície do CASE technológie nie sú zanedbateľné, súčasne je aj rýchly vývoj metód, metodológií a tiež každá úspešná softvérová spoločnosť hromadí skúsenosti z realizácie softvérových projektoch vo forme vlastných postupov. Tieto su často modifikáciou všeobecne známych metód a metodológií implementovaných v CASE systémoch.

Pre návratnosť prostriedkov vložených do CASE technológie sa ukazuje kritickou možnosť konfigurácie CASE systémov podľa potrieb riešiteľského tímu a podľa vlastností cieľového vyvíjaného systému. Zaradenie efektívnych tradičných alebo aj nových modelovacích nástrojov, semiformálnych - ako je napr. DFD, alebo formálnych špecifikačných jazykov by nemalo byť obmedzením pri použití CASE systémov. Predpokladom je, aby tieto systémy boli konfigurovateľné v implementovanej metodológii, metódach aj nástrojoch.

4. ZÁVER

Separácia toku údajov, toku riadenia a akcií, ktoré reprezentujú transformácie funkcionálneho a procedurálneho charakteru je jedným z predpokladov takého návrhu zložitých softvérových systémov, ktorý zabezpečuje ich odolnosť voči chybám a flexibilitu ich budúceho vývoja.

Samotné modelovanie softvérových systémov je v tomto smere nadstavbou, ktorej odôvodnenosť je daná jednak praktickými potrebami návrhárov a jej užitočnosť v súčasnej dobe je určená najmä v spätnej väzbe, ktorú poskytuje integračným snahám v oblasti špecifikačných [4] a multiparadigmatických implementačných jazykov [2]. Zložitosť súčasných softvérových systémov je možné zvládnuť iba integrovaným prístupom, a v tomto smere majú nezastupiteľnú úlohu jazyky pre modelovanie systémov, ako je napr. UML, ktorý podporuje myšlienku separácie jednotlivých stránok systému na úrovni jeho objektového modelu a tým určuje základnú kvalitu údržby a ďalšieho vývoja systému.

Preto hľadanie aj čiastkových kombinovaných prístupov k modelovaniu na praktickej úrovni návrhových systémov má veľký význam z hľadiska integrácie jazykov a softvérových architektúr v budúcnosti, ktorá by na jednej strane zabezpečila stabilitu softvérových systémov, na druhej strane umožnila ich flexibilitu a z pozície konštruktéra použitie rôznorodých prístupov systematickým spôsobom, primeraným na riešenie konkrétnej stránky softvérového systému.

LITERATÚRA

[1] Armstrong, B., Brown, M.F.: PowerBuilder 9: Advanced Client/Server Development, SAMS; 2004, ISBN: 0672325004.

[2] Kollár, J.: Control-driven Data Flow, Journal of Electrical Engineering, 51(2000), No.3-4, pp.67-74.

[3] Young, M. – Johnson, B. – Skibo, C.: Inside Microsoft Visual Studio .NET 2003. Microsoft Press 2003, ISBN: 0735618747.

[4] Novitzká, V.: Mathematical language in programming, Acta Electrotechnica et Informatica, 3, 3, 2003, pp. 31-35, ISSN 1335-8243.

[5] Hollinqworth, J. – Gustavson, P. – Swart, B. – Cashman, M.: Borland C++Builder 6 Developer's Guide. Sams 2002, ISBN 0672324806.

[6] Rumbaugh, J. – Jacobson, I. – Booch, G.: The Unified Modeling Language reference manual.

Addison-Wesley Object Technology Series, 1998, ISBN 0-201-30998-X.
[7] Sommerville, I.: Software Engineering (6th Edition). Addison-Wesley Pub Co. 2004 ISBN 0321210263.
[8] Paradigm Plus. Methods Manual. ProtoSoft Inc., 1995.
[9] Kollár, J., Samuelis, L., Rajchman, P.: Notes on Experience of Transforming Distributed Learning Materials into SCORM Standard Specifications, Information and Security, Vol.14, 2004, ISSN 1311-1493, pp. 81-86.

BIOGRAPHY

Zdeněk Havlice was born in 14.02.1958. In 1982 he graduated (MSc.) with honors at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice. He defended his PhD. in the field of visual programming and user interface design in 1991; his thesis title was: "Design of User Interface for Dialogue Systems". Since 1999 he is working as a associated professor at the Department of Computers and Informatics. His scientific research is focusing on the area of special languages, compilers, dialog prototyping, CASE systems, software methodologies, methods and tools.

Ján Kollár was born in 1954. He received his MSc. summa cum laude in 1978 and his PhD. in Computing Science in 1991. In 1978-1981 he was with the Institute of Electrical Machines in Košice. In 1982-1991 he was with the Institute of Computer Science at the University of P.J. Šafárik in Košice. Since 1992 he is with the Department of Computers and Informatics at the Technical University of Košice. In 1985 he spent 3 months in the Joint Institute of Nuclear Research in Dubna, Soviet Union. In 1990 he spent 2 month at the Department of Computer Science at Reading University, Great Britain. He was involved in the research projects dealing with the real-time systems, the design of (micro) programming languages, image processing and remote sensing, the dataflow systems, the educational systems, and the implementation of functional programming languages. Currently the subject of his research is the implementation of multi-paradigmatic languages.

Vladimír Chladný was born in 1976. He received his MSc. with honors in 1999 and he is a PhD. candidate in Computing Science. Since 2003 he is with the Department of Computers and Informatics at the Technical University of Košice. In 2001-2002 he spent 10 months at Philips Research Labs. in Eindhoven, The Netherlands. He was involved in the research projects dealing with the scenario-based architecting, formal specification of the software lifecycle processes and integrated CASE systems development. Currently the subject of his research is the CASE tools support for different software technologies.