PRIMERJAVA OBJEKTNO-ORIENTIRANIH METOD ANALIZE

Marjan Heričko, Ivan Rozman, Ivan Lah
Tehniška Fakulteta Maribor
Smetanova 17, 62000 Maribor

Povzetek

V prispevku primerjamo štiri uveljavljene objektno-orientirane metode analize, ki jih predlagajo Bailin, Shlaer & Mellor, Hayes & Coleman in Coad & Yourdon. Predvsem nas zanimajo predlagana pravila zapisa in postopki posameznih metod. Prav tako raziščemo, katere principe objektne orientiranosti je v obravnavanih metodah možno direktno uporabiti in predstaviti. Slednjič primerjamo objektno-orientirane metode analize s strukturno analizo.
Obravnavamo tudi možnost uporabe konvencionalnih orodij CASE v procesu objektno-orientiranega razvoja. Na koncu podamo svoje razloge in priporočila za izbiro najprimernejše objektno-orientirane metode analize.

Abstract

In this paper, the comparison of the four well-known object-oriented analysis approaches, proposed by Bailin, Shlaer & Mellor, Hayes & Coleman and Coad & Yourdon, is done. Mostly we are interested in the notation and the process they include and in the major features of the object-oriented paradigm that are addressed directly in the proposed notations. Also object-oriented analysis methods are compared to the structured analysis.
Possibilities of using conventional CASE tools in the object-oriented development process are considered. At the end our arguments and proposals for selecting the most appropriate object-oriented analysis method are given.

■ ■ ■

1. UVOD

Koncepti objektno-orientiranega pristopa so dosegli že takšno stopnjo razvoja, da pokrivajo oz. obsegajo celoten življenjski cikel informacijskih sistemov. Od sedemdesetih let se je težišče raziskav preneslo od uvajanja do zgodnejših faz razvoja, vključno s fazo analize sistemskih in programskih zahtev. To lahko razumemo tudi kot posledico dobro definirane tehnologije programiranja, kar potrjuje razvoj mnogih objektno-orientiranih programskih jezikov (C++, Smalltalk, ObjectPascal, CLOS...).

Objektno-orientirane metode raziskujemo z namenom, da bi ugotovili, katere metode in orodja so najprimernejši za podporo razvoju informacijskih sistemov. Preizkušamo, ali bi bilo mogoče uporabiti neke vrste delnega oziroma hibridnega objektno-orientiranega razvojnega procesa, ki bi na primer vključeval funkcionalno analizo, objektno-orientirano načrtovanje in implementacijo. Razen tega sta pomembni prednosti objektno-orientiranega pristopa možnost ponovne uporabe in lažje, enostavnejše vzdrževanje programske kode. Dodatno nas je k raziskavam objektne orientacije vodila potreba po sodobnih uporabniških vmesnikih.

Osnovni problem, s katerim se srečujemo v procesu razvoja programske opreme, je obvladovanje in nadzor kompleksnosti ter pogojne razumljivosti. Medtem ko funkcionalna analiza temelji na abstrakciji procesov, objektno-orientirani pristop nudi in zagotavlja nekatere dodatne in koristne, predvsem svojstvene principe. Kompleksnost obvladuje z naslednjimi principi: abstrakcija, ograjevanje, dedovanje in komunikacija s sporočili.

Upoštevanje in uporaba teh principov za obvladovanje kompleksnosti je le eden izmed kriterijev, ki smo jih uporabili v naši primerjavi. Preučili smo tudi pravila zapisa, postopek analize, pokrivanje vidikov sistema, prehod iz analize v načrtovanje in podporo orodij CASE.

Ker ne obstaja splošno soglasje o definiciji objektno-orientirane metode analize, podajmo najprej svojo definicijo ter se tako izognimo možnim nesporazumom:

Objektno-orientirana metoda analize se usmerja predvsem na modeliranje problemskega področja z objekti in njihovimi atributi. Nato doda funkcije, ki lahko delujejo le v povezavi z lastnostmi objekta, h kateremu spadajo. Odgovornost za neko funkcijo pripada enemu samemu objektu, ki lahko z drugi-

mi objekti komunicira prek pošiljanja sporočil in dogodkov.

Kot vidimo, naša definicija ni tako stroga kot tista, ki jo zasledimo v (Coad,90). Slednja namreč zahteva eksplicitnost funkcij, dedovanja in klasifikacijskih struktur, sicer metodo opredeljuje kot informacijsko modeliranje. Metode analize, ki smo jih izbrali na osnovi subjektivnih meril, so le najbolj tipični predstavniki obsežne množice objektno-orientiranih metod analize, ki so se pojavile v zadnjih letih. Običajno naletimo na izbrane metode tudi v publikacijah, ki se ukvarjajo s podobno tematiko, npr. (Monarchi,92; Fichman,92).

2. PREGLED IZBRANIH METOD

Bailin-ova metoda specificiranja (Bailin,89)
Poudarek je na vsebini entitet in nič več na transformaciji vhodov v izhode. Osnovne aktivnosti so usmerjene k identifikaciji entitet, ki jih nadalje razgradimo v podentitete in/ali funkcije. Funkcije združujemo po principu povezanosti z istimi podatki. Kot rezultat dobimo specifikacije, sestavljene iz hierarhije entitetnih diagramov toka podatkov, množice entitetno relacijskih diagramov in entitetnega slovarja.

Shlaer-jev pristop k analizi problemskega prostora (Shlaer,88;89)
Metoda temelji na semantičnem podatkovnem modeliranju. Usmerja se predvsem na izdelavo prepričljive in verodostojne kopije stvarnosti. Pristop zahteva izgradnjo treh tipov formalnih modelov in sicer informacijskega modela, množice modelov stanj in množice procesnih modelov. Vsekakor metoda kombinira že dalj časa poznane in uporabljane modele, z natančno predpisanimi pravili integracije.

Hayes-ova objektno-orientirana analiza (Hayes,91)
Uporablja modele z natančno semantiko in konsistentno tehniko za izvajanje analize problemskega prostora. Matematična osnova urejene predikatne logike z vgrajenimi tipi omogoča preverjanje skladnosti modelov. Za opisovanje operacij se uporabljajo formalne specifikacije, podobne jeziku VDM. Model objektnih struktur povezuje objektni model z dinamičnimi in funkcionalnimi modeli.

Coad-ova analiza (Coad,90)
Metoda skuša premostiti prepad med diagrami toka podatkov in entitetno relacijskimi diagrami. Vpeljana pravila zapisa objektnih diagramov kombinirajo bistvene koncepte obeh grafičnih predstavitev ter uvajajo dodatne zmožnosti. Metoda temelji na treh osnovnih načelih človeške organiziranosti: objekti in atributi, razredi in člani ter celote in deli. Osnovni cilj tega pristopa je boljše razumevanje problemskega prostora. Atribute objektov in pripadajoče funkcije teh lastnosti obravnavamo kot nedeljivo celoto.

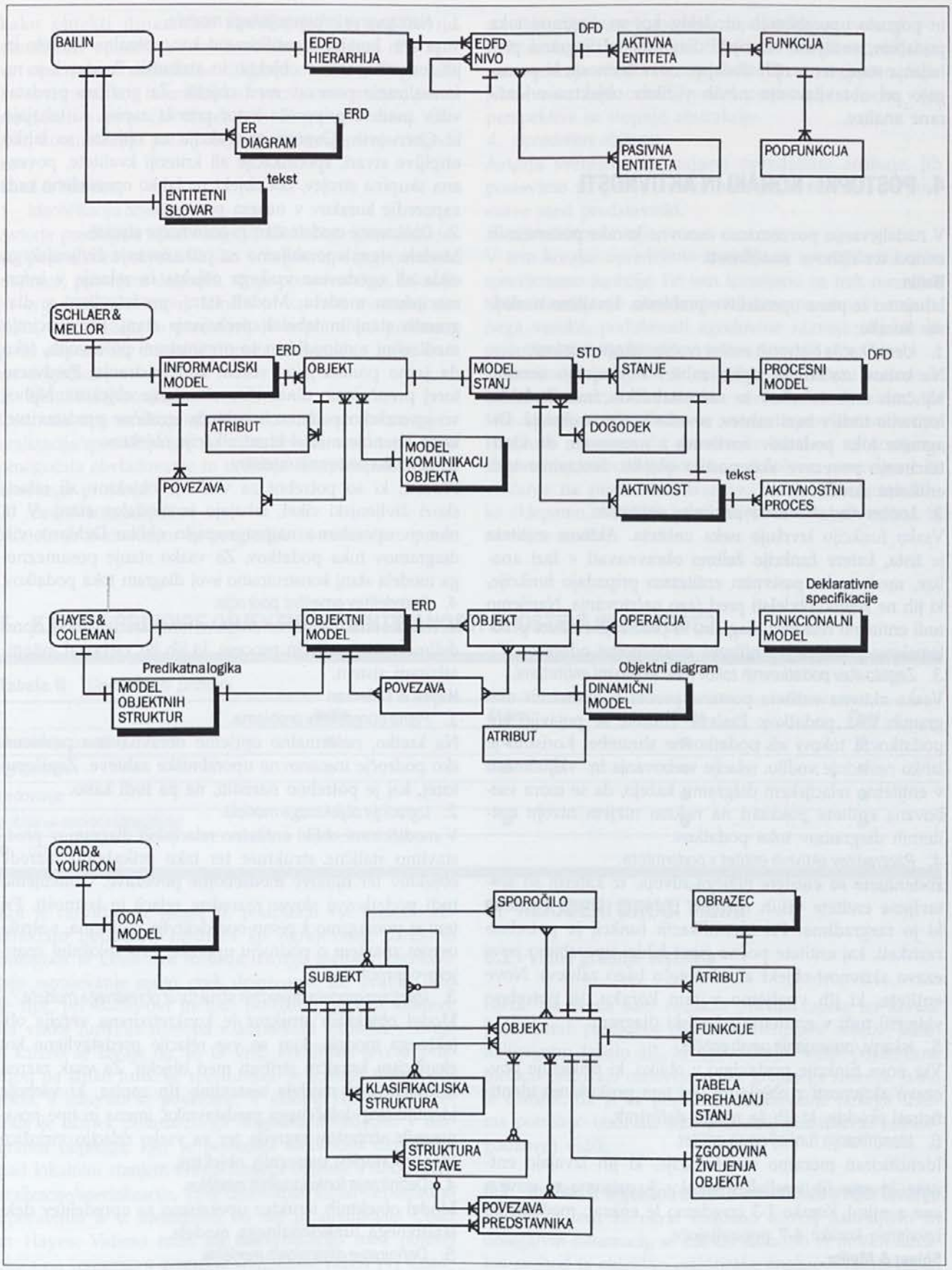
3. ELEMENTI IN PRAVILA ZAPISA

Slika 1 prikazuje elemente in njihove povezave v posameznih metodah. Oznake nad okvirčki pomenijo uporabljeno grafično ali tekstovno predstavitev, medtem ko osenčeni okvirčki poudarjajo pomembne rezultate obravnavanih metod.
Kot je razvidno iz tabele I in slike 1, kombinirajo obravnavane metode na različne načine značilnosti poznanih

Tabela I *Pregled pravil zapisa v metodah*

Metoda	Bailin	Shlaer	Hayes	Coad
Grafične predstavitve				
Diagrami toka podatkov	entitetni diagram toka podatkov	procesni modeli		
Entitetno relacijski diagrami	entitetno relacijski diagrami	informacijski modeli	objektni model	
Diagrami prehajanja stanj		modeli stanj	objektni diagrami	
Objektni komunikacijski model		●		
Objektni model				●
Besedilo				
Opredelitev problema	●		●	●
Tekstovni opisi	entitetni slovar	aktivnosti		obrazec za objekt
Formalne specifikacije			model objektnih struktur, funkcionalni model	

● - uporablja



Slika 1: *Elementi obravnavanih metod*

in pogosto uporabljanih modelov, kot so diagrami toka podatkov, entitetno relacijski diagrami in diagrami prehajanja stanj, ter v njih dodajajo nove lastnosti, ki pomagajo pri obravnavanju novih vidikov objektno-orientirane analize.

4. POSTOPKI - KORAKI IN AKTIVNOSTI

V nadaljevanju povzemamo osnovne korake posameznih metod ter njihove značilnosti:

Ballin

Izhajamo iz pisne opredelitve problema. Izvajamo naslednje korake:

1. *Identifikacija bistvenih entitet problemskega prostora.*

Na osnovi začetnega opisa zahtev oblikujemo seznam ključnih samostalnikov in samostalniških fraz. Te lahko lociramo tudi v bazi zahtev, seveda le, če ta obstaja. Diagrame toka podatkov narišemo z namenom, da identificiramo povezave aktivnosti z objekti. Sestavimo tudi entitetni slovar.

2. *Ločitev med aktivnimi in pasivnimi entitetami.*

Vsako funkcijo izvršuje neka entiteta. Aktivna entiteta je tista, katere funkcije želimo obravnavati v fazi analize, medtem ko pasivnim entitetam pripadajo funkcije, ki jih ne želimo obdelati pred fazo načrtovanja. Narišemo tudi entitetno relacijski diagram, ki prikazuje entitete problemskega prostora ter njihove medsebojne odnose.

3. *Zagotovitev podatkovnih tokov med aktivnimi entitetami.*

Vsaka aktivna entiteta postane proces v entitetnih diagramih toka podatkov. Pasivne entitete se pojavijo kot podatkovni tokovi ali podatkovne shrambe. Koristno je lahko naslednje vodilo: relacije vsebovanja in vključenosti v entitetno relacijskem diagramu kažejo, da se mora vsebovana entiteta pokazati na nekem nižjem nivoju entitetnih diagramov toka podatkov.

4. *Razgraditev aktivnih entitet v podentitete.*

Podentitete so entitete nižjega nivoja, iz katerih so sestavljene entitete višjih nivojev. Funkcije izvaja entiteta, ki jo razgradimo. Pri identifikaciji funkcij je potrebno raziskati, kaj entitete počno (spet lahko uporabimo povezavo aktivnost-objekt ali obstoječo bazo zahtev). Nove entitete, ki jih vpeljemo v tem koraku, je potrebno vključiti tudi v entitetno relacijski diagram.

5. *Iskanje/preverjanje novih entitet.*

Vse nove funkcije postavimo v obliko, ki prikazuje povezavo aktivnosti z objektom. Pri tem poskušamo identificirati objekte, ki jih še nismo definirali.

6. *Identifikacija funkcij novih entitet.*

Identificirati moramo vse funkcije, ki jih izvajajo entitete, ki smo jih vpeljali v koraku 5, oziroma so povezane z njimi. Korake 1-3 izvedemo le enkrat, medtem ko izvajamo korake 4-7 ponavljajoče.

Shlaer & Mellor

Model izdelamo v naslednjem zaporedju:

1. *Načrtovanje informacijskega modela.*

V prvem koraku identificiramo konceptualne entitete in jih formaliziramo v objektih in atributih. Poudarek je na formalizaciji povezav med objekti. Za grafično predstavitev modela priporoča avtor pravila zapisa, ki izhajajo iz Chen-ovih. Osnova abstrakcije za objekte so lahko otipljive stvari, specifikacije ali kriteriji kvalitete, povezana skupina strojev, kot objekt pa lahko opredelimo tudi zaporedje korakov v nekem procesu.

2. *Oblikovanje modela stanj za posamezne objekte.*

Modele stanj uporabljamo za prikazovanje življenjskega cikla ali zgodovine vsakega objekta in relacije v informacijskem modelu. Modeli stanj, predstavljeni v diagramih stanj in tabelah prehajanja stanj, komunicirajo med seboj z dogodki in so organizirani po nivojih, tako da jasno ponazarjajo sistem komuniciranja. Predvsem torej preučujemo dinamično obnašanje objektov. Njihovo interakcijo podamo s pomočjo grafične predstavitve, ki se imenuje model komunikacije objektov.

3. *Izdelava procesnih modelov.*

Procesi, ki so potrebni za vodenje objektov ali relacij skozi življenjski cikel, izhajajo iz modelov stanj. V ta namen uporabimo najpreprostejšo obliko DeMarco-vih diagramov toka podatkov. Za vsako stanje posameznega modela stanj konstruiramo svoj diagram toka podatkov.

4. *Opredelitev omejitve področja.*

V tem koraku določimo meje avtomatizacije, torej opredelimo informacije in procese, ki jih bo vseboval avtomatizirani sistem.

Hayes & Coleman

1. *Pisna opredelitev problema.*

Na kratko, neformalno opišemo obravnavano problemsko področje in osnovne uporabniške zahteve. Zapišemo torej, kaj je potrebno narediti, ne pa tudi kako.

2. *Izgradnja objektnega modela.*

V modificirani obliki entitetno relacijskih diagramov predstavimo statične strukture ter tako prikažemo razrede objektov ter njihove medsebojne povezave. Oblikujemo tudi podatkovni slovar razredov, relacij in lastnosti. Pri tem si pomagamo s pisno opredelitvijo problema, s strokovnim znanjem o področju uporabe ter s splošnim znanjem o problemu.

3. *Izpeljava modela objektnih struktur iz objektnega modela.*

Model objektnih struktur je konkretizirana verzija objektnega modela, kjer so vse relacije predstavljene kot eksplicitni kazalčni atributi med objekti. Za vsak razred iz objektnega modela sestavimo tip zapisa, ki vsebuje identifikator določenega predstavnika, imena in tipe posameznih atributov razreda ter za vsako relacijo množico identifikatorjev ustreznih objektov.

4. *Definiranje funkcionalnih modelov.*

Model objektnih struktur uporabimo za opredelitev deklarativnega funkcionalnega modela.

5. *Definiranje dinamičnih modelov.*

Pozornost posvetimo obnašanju modela ter specificiramo,

kako objekti dinamično sodelujejo, da zagotovijo obnašanje, opisano s funkcionalnim modelom. Pri tem za vsak razred definiramo diagram objektov, ki ga lahko smatramo kot formalizirano verzijo avtomatov, predstavljenih s pomočjo Harel-ovih diagramov stanj. Objekti komunicirajo prek pošiljanja in sprejemanja dogodkov.

Coad & Yourdon

1. Identifikacija objektov.

Avtorja predlagata opazovanje problemskega prostora, kjer so potencialni objekti lahko strukture, drugi sistemi, naprave, pomnjeni dogodki, vloge posameznikov, lokacije in organizacijske enote. Izločiti in zavrniti moramo objekte, ki odražajo in predstavljajo nepotrebno pomnenje in nepotrebne funkcije, izpeljane rezultate in enkratne dogodke.

2. Identifikacija struktur.

Osnovna tipa struktur sta klasifikacijska struktura (generalizacija/specializacija) ter združitvena struktura. Obe omogočata obvladovanje in urejanje kompleksnosti problemskega prostora.

3. Identifikacija subjektov.

Subjekt je mehanizem za pregledno in razumljivo predstavitev diagramov modela. Upoštevati je namreč potrebno, da so bralčeve zmožnosti hkratnega razmišljanja, razumevanja in obravnave večih stvari omejene. Subjektni nivo zato celotni model predstavlja z neke višje perspektive in stopnje abstrakcije.

4. Opredelitev atributov

Avtorja svetujeta, da najprej opredelimo atribute, jih postavimo na pravo mesto, nato pa definiramo še povezave med predstavniki.

5. Opredelitev funkcij.

V tem koraku opredelimo funkcije in sporočila ter nato specificiramo funkcije. Pri tem temeljimo na treh osnovnih tipih klasifikacije obnašanja in sicer na osnovi trenutnega vzroka, podobnosti zgodovine razvoja objekta in podobnosti funkcij.

Iz podanega pregleda je razvidno, da se vse metode v začetku usmerjajo na objekte, atribute in strukture, ter šele kasneje na funkcije in medsebojno komunikacijo. Poudariti moramo, da navedenih korakov ne izvajamo zaporedno, temveč je potrebno določeno ponavljanje in vračanje na predhodne faze oz. korake. Tudi zato lahko sklepamo, da imajo postopki obravnavanih metod še razvojen značaj.

5. KATERE PRINCIPE OBJEKTNE ORIENTIRANOSTI UPOŠTEVAJO METODE?

Tabela II *Upoštevani principi*

	Bailin	Shlaer	Hayes	Coad
abstrakcija podatkov	●	●	●	●
ograjevanje	●	●	●	●
dedovanje			●	●
pošiljanje sporočil (dogodkov)		●	●	●

● - uporablja

Kot je razvidno iz tabele II, podpirajo vse metode abstrakcijo podatkov, medtem ko ograjevanje direktno omogoča le Coad-ova metoda. Druge metode zagotavljajo ograjevanje samo prek dogovorov oz. pravil. Tako je npr. pri Bailin-ovi metodi potrebno ugoditi zahtevi, da vsaka funkcija nastopa v kontekstu entitete, v zvezi s katero se izvaja oz. jo ta vrši. Pri Shlaer-jevem pristopu pa lahko funkcije spreminjajo le atributi tistega objekta, s katerim so povezane. V sklopu Hayes-ove metode je učinek posameznega dogodka opredeljen v diagramu objektov, kjer je posledica aktivnosti definirana nad lokalnim stanjem enega samega objekta. Princip generalizacije/specializacije, torej dedovanja, lahko neposredno uporabimo le z metodama, ki sta ju predlagala Coad in Hayes. Vidimo tudi, da komunikacijo med objekti običajno izrazimo s pomočjo dogodkov, razen pri Coadu, ki v ta namen uporablja sporočila.

6. NEKATERI DRUGI VIDIKI

6.1 Vidiki gradnje sistema

Glede na to, da smo raziskali pravila zapisa ter korake posameznih predlogov objektno-orientirane analize, lahko oblikujemo tabelo III, ki opredeljuje, kako posamezne metode pokrivajo osnovne vidike grajenega sistema. Jasno je namreč, da je za popolno razumevanje nekega sistema potrebno obdelati tako procesni, podatkovni kot dogodkovni vidik.

6.2 Prehod iz objektno orientirane analize v načrtovanje

Čeprav o tem zaenkrat nimamo dovolj zanesljivih in dosegljivih informacij, se zdi, da navadno ni možen direkten prehod iz objektno-orientirane analize v načrtovanje. Potrebno je opraviti še nekaj dodatnega dela. Npr.: za

Tabela III *Komponente, ki pokrivajo določen vidik sistema*

vidik	Bailin	Shlaer	Hayes	Coad
podatkovni	entitetno relacijski diagram	informacijski model	objektni model*	objektni model*
dogodkovni	-	model stanj, objektni model komunikacij	objektni diagram	diagram prehajanja stanj
procesni	funkcije, podfunkcije	procesi	aktivnosti	funkcije

* Hayes-ov in Coad-ov objektni model se znatno razlikujeta

Bailin-ovo metodo so razvili generator načrtovanja, ki na osnovi rezultatov analize (hierarhije entitetnih diagramov toka podatkov) oblikuje ADA-orientirane objektne diagrame. Pred uporabo generatorja pa mora razvijalec entitetne diagrame toka podatkov dopolniti z nekaterimi dodatnimi informacijami. Drugače pa je pri Coad-ovi metodi, kjer postanejo rezultati analize integralni del večkomponentnega objektno-orientiranega načrtovanja, kot ga predlaga (Coad,91).

Vsekakor velja ugotovitev, da je za konsistenten prehod iz objektno-orientirane analize v načrtovanje potrebno zagotoviti enotno predstavitev za obe fazi razvoja.

6.3 Objektno-orientirana analiza v primerjavi s strukturno analizo

V sklopu strukturne analize analiziramo sistem s funkcionalnega vidika, ki poudarja transformacijo vhodov v izhode ter funkcionalno razgradnjo. Medtem ko pomeni strukturna analiza predvsem "top-down" strategijo, lahko rečemo, da je objektno-orientirana analiza predvsem "bottom-up" pristop. Kot rezultat strukturne analize je struktura modela sistema hierarhija funkcij; pri objektno-orientirani analizi izkazuje objektni model nehierarhično topologijo objektov. Sam prehod iz strukturne analize v načrtovanje, je navkljub strategijam kot sta transformacijska in transakcijska analiza zahtevno opravilo, saj preslikava ni izomorfna. Slednje lahko v sklopu objektno-orientiranega razvoja zagotovimo z enovito predstavitvijo.

Omeniti velja še ključno vprašanje, ki se postavlja v zvezi s primerjavo obeh pristopov in sicer: Kateri način razmišljanja je človeku bližji? Ali o objektih ali o funkcijah? Sami se ne pridružujemo zagovornikom prve ali druge možnosti, saj podobno kot (Loy,90) menimo, da to vprašanje najverjetneje nikoli ne bo razrešeno.

6.4 Podpora orodij CASE

Menimo, da je razvoj objektno-orientirane analize precej odvisen od razvoja in uporabe ustreznih orodij in metod, saj je ta tehnologija preveč kompleksna, da bi bila splošno razširjena in uporabljana brez ustrezne računalniške podpore.

Kolikor je nam znano, je želja po tem, da bi orodje CASE služilo ne le samo kot risarsko orodje, zadovoljena le za Coad-ovo metodo, ki jo podpira orodje OOATool™. Jasno je, da ni smiselno uporabiti Hayes-ove metode, saj ne obstaja ustrezna računalniška podpora, ki bi vključevala tudi preverjanje skladnosti. Za preostali metodi (Bailin in Shlaer) pa poglejmo možnosti uporabe nekaterih konvencionalnih orodij (POSE®, IEW®, Excelerator® and Analyst/Designer Toolkit™). Tabeli I in IV nam pomagata odgovoriti na vprašanje: ali lahko pri izvajanju objektno- orientirane analize uporabimo konvencionalna orodja CASE? Omenjena orodja lahko uporabimo kot pripomočke za risanje, medtem ko je za sintaksno in semantično preverjanje ter avtomatiziran prehod v načrtovanje potrebno zagotoviti dodatno programsko podporo oz. opremo. V ta namen mora orodje CASE zagotoviti vmesnik z zunanjimi sistemi, tako da lahko informacije prenašamo v ustrezno okolje CASE in iz njega. To tematiko obravnava tudi članek (Heričko,92).

Tabela IV *Funkcije, ki jih zagotovljajo orodja CASE*

orodja CASE	ERD	DFD	STD	Import/Export
POSE®	●	●		●
IEW® Analysis Workstation	●	●		●
Excelerator® IS	●	●	●	●
Analyst/Designer Toolkit™	●	●	●	●

7. ZAKLJUČKI

Na osnovi rezultatov primerjave po različnih kriterijih, nanizanih v prejšnjih poglavjih, lahko podamo in utemeljimo razloge, ki nas vodijo pri izbiri ustrezne oz. primerne metode objektno-orientirane analize.

Čeprav lahko Bailin-ovo metodo obravnavamo kot objektno-orientirano, v njej dedovanja ni možno direktno uporabiti in izraziti. Prav tako ni prave povezave med entitetnimi diagrami toka podatkov in entitetno relacijskimi diagrami. Razen tega opažamo tudi določeno podvajanje informacij med obema modeloma.

Menimo, da vpeljava formalnih specifikacij, kot je to izvedeno pri Hayes-ovi metodi, ni posebej priporočljiva in koristna v smislu komunikacije z uporabniki in naročniki. Uspešnost izvajanja le-te pa ima seveda precejšen vpliv na kakovost izvedbe analize. Iz tega sledi, da za nas ostajata zanimiva le pristopa, ki ju predlagata Shlaer in Coad. Kot vidimo, je uporaba Coad-ove metode pogojena z ustreznim orodjem CASE, medtem ko si lahko v primeru, ko uporabimo Shlaer-jev pristop, pomagamo z nekaterimi konvencionalnimi orodji. Sklepamo torej lahko, da je v primeru, ko naletimo na pomanjkanje ustreznega orodja, smiselno uporabiti Shlaer-jevo metodo, ne glede na to, da ne zagotavlja direktnega prehoda oz. preslikave v načrtovanje in ne oziraje se na nekatera mnenja, da ta metoda ne predstavlja resničnega objektno-orientiranega pristopa.

Coad-ova metoda je najverjetneje tista, ki jo bomo v praksi najpogosteje uporabljali (Heričko,92a). Naš zaključek temelji na dejstvu, da ta metoda izmed vseh obravnavanih najbolje zajema potrebe in zahteve glede principov objektne orientiranosti, zagotavlja enostaven prehod v načrtovanje in nenazadnje, obstaja tudi ustrezno orodje CASE, ki jo podpira.

V prihodnosti pričakujemo skladno objektno-orientirano metodologijo za celoten življenjski cikel informacijskih sistemov. V ta namen je potrebno obstoječe metode preizkusiti v praksi in sicer za različne tipe sistemov. Tudi na osnovi ugotovljenih pomanjkljivosti in problemov, kot jih npr. zasledimo v (Aksit,92), lahko odgovorimo na vprašanji, na kateri je potrebno poiskati vsaj okvirna odgovora : Ali je objektno-orientirana tehnologija primerna za vse vrste sistemov oz. za katere vrste sistemov je ta pristop še posebej primeren.

8. LITERATURA

(Aksit,92) Aksit M., Bergmans L., Obstacles in Object-Oriented Software Development, ACM SIGPlan Notices, OOPLSA'92, Conference Proceedings, Vol. 27, No. 10, 1992, pp. 341-358

(Bailin,89) Bailin S. C., An Object-Oriented Requirements Specification Method, Communications of the ACM, Vol. 32, No. 5, 1989, pp. 608-623

(Coad,90) Coad P., Yourdon E., Object-Oriented Analysis, Prentice Hall, New Jersey, 1990

(Coad,91) Coad P., Yourdon E., Object-Oriented Design, Prentice Hall, New Jersey, 1991

(Hayes,91) Hayes F., Coleman D., Coherent models for Object-Oriented Analysis, OOPSLA'91, Sigplan Notices, Vol. 26, No. 11, Phoenix, Arizona 1991, pp. 171-183

(Fichman,92) Fichman R. G., Kemerer C. F., Object-Oriented and Conventional Analysis and Design Methodologies, IEEE Computer , Vol. 25, No. 10, 1992, pp. 22-39

(Heričko,92) Heričko M., Lah I., Györkös J., Rozman I., Connection of the Development and the Target Environment in the Computer Aided Software Engineering, Proceedings of the First Electrotechnical and Computer Science Conference ERK'92, Volume B, Slovenia Section IEEE, Portorož, Slovenia 1992, pp. 99-102

(Heričko,92a) Heričko M., Rozman I., Györkös J., Lah I., Comparison of Object-Oriented Analysis Methods, Proceedings of the International AMSE Conference "Signals, Data, Systems", AMSE Press, Vol. 2, Calcutta (India), 1992, pp. 67-76

(Loy,90) Loy P. H., A Comparision of Object-Oriented and Structured Development Methods, ACM Software Engineering Notes, Vol. 15, No. 1, 1990, pp. 44-48

(Monarchi,92) Monarchi D. E., Puhr G. I., A Research Typology for Object-Oriented Analysis and Design, Communications of the ACM, Vol. 35, No. 9, 1992, pp. 35-47

(Shlaer,88) Shlaer S., Mellor S. J., Object-Oriented Systems Analysis: Modeling the World in Data, Prentice Hall, 1988

(Shlaer,89) Shlaer S., Mellor S.J., An Object-Oriented Approach to Domain Analysis, ACM Software Engineering Notes, Vol. 14, No. 5, 1989, pp. 66-77