PRIMERJAVA METODOLOGIJ OOAD IN OOSE ZA RAZVOJ INFORMACIJSKEGA SISTEMA

Milan Črv
Klinični center, Informacijski center, Zaloška c. 2, 1525 Ljubljana

Povzetek

Objektno orientiran pristop k razvoju informacijskih sistemov prispeva k boljšemu razumevanju poslovnega sistema in olajša razvoj rešitev. Obstaja veliko število objektno orientiranih metodologij za analizo in načrtovanje sistema, ki imajo navadno več skupnih lastnosti kot pa razlik. Osnovne uporabljene tehnike so si precej podobne, uvajajo pa se tudi novi pristopi k izgradnji različnih modelov sistema ter različni načini zapisovanja in predstavitve rezultatov. Namen prispevka je predstavitev in medsebojna primerjava glavnih značilnosti dveh razširjenih objektno orientiranih metodologij (OOAD in OOSE).

Abstract

Object-oriented information systems development helps understanding the problem domain better, and building the applications easier. Many object-oriented analysis and design methodologies have been published. Upon close examination, their similarities may be greater than their differences. Although the basic techniques are the same, each methodologist has a habit of altering them, introducing some new modeling concepts and a new notation. This paper introduces two popular methodologies (OOAD and OOSE) and sets out to identify the main similarities and differences between them.

1. Uvod

Prisotnost objektno orientiranih pristopov na področju razvoja informacijskih sistemov je več kot očitna. Poslovni sistemi so prisiljeni slediti hitremu tehnološkemu razvoju s prilagajanjem poslovnega procesa in njegove informacijske podpore, če želijo ohraniti ali celo izboljšati konkurenčno sposobnost. Zahteve po hitrem prilagajanju spremembam znotraj poslovnih sistemov, kakor tudi spremembam v njihovem okolju, so odkrile pomanjkljivosti tradicionalnih pristopov. Cilj in bistvo objektno orientiranih pristopov je prav poenostavitev izgradnje kompleksnih informacijskih sistemov. Ponovna uporaba že obstoječih, preizkušenih objektov bistveno zmanjša čas razvoja in zviša kvaliteto rešitev. Možnost spreminjanja lastnosti posameznih objektov brez vpliva na druge objekte pa omogoča lažje vzdrževanje rešitev.

2. Metodologija OOAD

Metodologija OOAD[1] (Booch, 1994a) zajema štiri glavne aktivnosti za opis sistema. Logična struktura se opiše s pomočjo diagramov razredov in objektov, fizična struktura pa z diagrami modulov in procesov. Za opis dinamike razredov se uporabljajo diagrami stanj, za opis dinamike objektov pa časovni diagrami. Osnovni ideji pristopa sta postopnost izgradnje in ponavljajoči se proces znotraj posameznih faz. Metodologija obravnava celoten proces razvoja z makro (makroproces) in mikro (mikroproces) vidika.

2.1. Makroproces

Z makro vidika je proces razvoja sistema razdeljen na naslednje faze (Booch, 1996, str.77):

- konceptualizacijo, kjer se opredeli okolje sistema, omejitve sistema in osnovne zahteve sistema;
- analizo, kjer se zgradi model sistema;
- načrtovanje, kjer se definira zgradba (arhitektura) sistema za implementacijo;
- razvoj, kjer se sistem implementira na podlagi modelov, zgrajenih v predhodnih fazah;
- vzdrževanje, kjer se upravljajo kasnejše dograditve in spremembe sistema.

[1] *Object-Oriented Analysis and Design.*

Znotraj posameznih faz makroprocesa se odvija ponavljajoč se mikroproces. Makroproces je tako ogrodje, ki kontrolira posamezne mikroprocese. Faze makroprocesa so podobne fazam v tradicionalnem modelu, razlika je le v odsotnosti precej strogega pristopa od zgoraj navzdol. Mikroproces pa je dograjevajoč in ponavljajoč se postopek, ki je precej bližji spiralnemu modelu razvojnega cikla sistema (van der Walt, 1994, str. 164).

2.2. Mikroproces

Glavni rezultati mikroprocesa so prototipi ter načrti za arhitekturo in implementacijo sistema. Koraki mikroprocesa s poudarkom na fazi analize so:

1. Identifikacija razredov in objektov. Identifikacija razredov in objektov je prvi korak v omejitvi področja problema. Rezultat prizadevanj je slovar abstrakcij, pri čemer razumemo abstrakcije kot bistvene karakteristike objektov, ki razlikujejo objekte med seboj. Slovar abstrakcij vsebuje poleg definicij razredov in objektov tudi mehanizme, ki opredeljujejo sodelovanje različnih objektov pri zagotavljanju obnašanja sistema na višjem nivoju.
   Aktivnost identifikacije razredov lahko razdelimo na naslednje korake:
   - kreiranje seznama kandidatnih razredov;
   - identifikacija operacij posameznih razredov, kajti za vsako obnašanje sistema morajo obstajati abstrakcije, ki sprožijo določeno obnašanje sistema in sodelujejo v njem;
   - preizkus izbora razredov na podlagi različnih scenarijev, ki opisujejo posamezno obnašanje sistema.
2. Opredelitev razredov in objektov. Namen je opredelitev obnašanja in identifikacija svojstev za abstrakcije, definirane v predhodni fazi. Obstajajo trije pristopi za opredelitev razredov in objektov, ki se med seboj dopolnjujejo:
   - Načrtovanje s scenariji. Načrtovanje s scenariji je pristop od zgoraj navzdol pri opredelitvi posameznih razredov. Na podlagi izbranih scenarijev in ustreznih abstrakcij se identificirajo pripadajoče operacije in svojstva posameznih razredov.
   - Analiza posameznih razredov. Analiza posameznih razredov je pristop od spodaj navzgor. Za vsak posamezen razred opredelimo njegove vloge in odgovornosti. Vlogo razumemo kot razlog, zaradi katerega je posamezen razred v povezavi z drugim razredom, odgovornost pa lahko opredelimo kot obnašanje objekta. Posamezni razred opišemo z elementarnimi operacijami, pri čemer poizkušamo zagotoviti ponovno uporabo že opredeljenih operacij za podobne vloge in odgovornosti.
   - Analiza skupnih lastnosti razredov. Na podlagi analize skupnih lastnosti se odločamo o poziciji razredov v hierarhiji abstrakcij. Skupne lastnosti posameznih razredov se tako navadno shranijo v nadrazredu.
3. Identifikacija povezav med razredi in objekti. Identifikacija povezav med razredi in objekti zajema opredelitev splošnih povezav med posameznimi razredi, ki omogočajo navigacijo med posameznimi objekti. Definirajo se tudi druge oblike povezav, npr. dedovanja (specializacija ali posplošenje) in agregacije[2]. Za vsako povezavo opredelimo vlogo posameznega razreda v povezavi in vrsto povezave. Opredeljene povezave preverimo na podlagi pripravljenih scenarijev za posamezno obnašanje sistema.
4. Implementacija razredov in objektov. Implementacija razredov in objektov je dejansko področje faze implementacije sistema, kot pove že ime samo. Vendar v tem koraku v fazi analize preverimo rezultate predhodnih korakov. Če ugotovimo določene pomanjkljivosti modela, se odločimo za ponovitev korakov mikroprocesa znotraj faze analize.

2.3. Sklepne ugotovitve

Z združitvijo obeh, med seboj različnih procesov (makroprocesa in mikroprocesa) v enoten razvojni proces, dosežemo:

- dobro voden in kontroliran proces razvoja sistema na podlagi ogrodja, ki nam ga nudi makroproces;
- dovolj svobode in inovativnosti znotraj posameznih faz razvoja sistema s ponavljanjem posameznih korakov mikroprocesa.

Poglejmo si še nekatere priporočljive napotke za uspešnejšo uporabo metodologije:

- v vseh fazah razvojnega cikla sistema je potrebno imeti jasno strateško in taktično sliko arhitekture sistema;
- z razvojem prototipov je možno enostavno preveriti ustreznost posameznih rezultatov analize in načrtovanja sistema;
- prototipi naj se zavržejo, ko so izpolnili namen, zaradi katerega so bili razviti;
- vzpostavljena naj bo prava mera formalizma, ki ne bo zavirala inovativnosti in bo omogočala zadostno kontrolo v celotnem razvojnem ciklu sistema;
- potrebno je opredeliti zunanje obnašanje sistema, pogoje in odgovore na nezaželjene dogodke;
- dokumentacija sistema ne sme biti namen razvoja sistema, ampak je le nujen, vzporeden proizvod.

[2] *Agregacija je opredeljena s povezavo 'je sestavljen iz'.*

3. Metodologija OOSE

Metodologija OOSE[3] (Jacobson et al., 1995) priporoča izgradnjo modelov analize in načrta sistema na osnovi zaporedja akcij med sistemom in njegovimi uporabniki, ki so opisane s posameznimi scenariji. Sistem, zgrajen na tak način, je bolj stabilen, uporaben in prilagodljiv spremembam in novim zahtevam (Jacobson, Christerson, Constantin, 1994, str. 247). Metodologija temelji na treh, med seboj različnih tehnikah:

- objektno orientiranem programiranju, ki prinaša v metodologijo osnovne koncepte objektne orientacije (ograjevanje, dedovanje, povezave med razredi in objekti);
- konceptualnem modeliranju, s katerim razumemo predvsem modeliranje podatkov, ki ga metodologija OOSE nadgrajuje z objektno orientiranimi sestavinami in možnostjo izražanja dinamičnega obnašanja sistema;
- načrtovanju blokov, ki omogoča lažje uvajanje dodatnih zahtev in vzdrževanje sistema.

3.1. Modeli sistema

Za obvladovanje kompleksnosti predlaga metodologija OOSE uporabo različnih modelov, ki opisujejo sistem z različnih vidikov:

1. Model zahtev sistema. V modelu zahtev sistema so zbrane funkcionalne zahteve z vidika uporabnikov sistema. Elementi modela zahtev sistema so:
   - Model scenarijev. Model scenarijev je enostaven način definiranja nosilcev vlog zunaj sistema in različnih primerov obnašanja sistema (scenarijev). Nosilec vloge predstavlja določeno vlogo, ki jo ima uporabnik sistema v določenem trenutku, lahko ga razumemo tudi kot razred, katerega primerki (objekti) so posamezni uporabniki. Ko uporabnik kot nosilec določene vloge uporablja sistem, izzove določeno obnašanje sistema, ki ga lahko opišemo s scenarijem. Opis scenarija lahko razumemo kot razred, vsako izvedbo pa primerek (objekt) razreda scenarijev. Posamezni scenariji opisujejo posamezno področje uporabe sistema, opisi vseh scenarijev pa predstavljajo opis celotne funkcionalnosti sistema. To nam omogoča, da lahko obravnavamo posamezne probleme neodvisno od drugih problemov. Za uveljavitev sprememb je potrebno le spremeniti model za posamezne nosilce vlog in pripadajoče scenarije. V scenarijih opredeljene funkcionalnosti sistema se strukturirajo v logičen, stabilen in od okolja implementacije sistema neodvisen model analize. Model načrta sistema pa je podroben model analize, prilagojen okolju implementacije sistema. Scenariji se implementirajo z generiranjem programske kode v modelu implementacije. V zadnji fazi nudijo dobro osnovo za testiranje posameznih podsistemov in izvedbo končnega testa sistema.
   - Opis uporabniških vmesnikov. Za podporo modela scenarijev je koristno razviti ustrezne uporabniške vmesnike. Prototipi uporabniških vmesnikov se v kombinaciji s simulacijo posameznih scenarijev izkažejo kot učinkovit način praktičnega preizkusa ustreznosti definiranih zahtev sistema.
   - Model področja sistema. Že v fazi definicije novega sistema se na podlagi modela scenarijev lahko zgradi model področja sistema, ki ga sestavljajo objekti z obravnavanega področja.
2. Model analize sistema (logični model). Model zahtev se podrobno razgradi v logični model sistema, ki nudi dovolj stabilno podlago za razvoj in vzdrževanje sistema. Informacijski prostor, v katerem se gradi model analize sistema, ima tri dimenzije. Informacijska dimenzija opredeljuje stanje sistema na podlagi informacij, ki se v sistemu hranijo. Dimenzija obnašanja opredeljuje obnašanje sistema, se pravi način spremembe stanja sistema. Prezentacijska dimenzija pa opisuje podrobnosti predstavitve sistema okolju. Znotraj tako opredeljenega informacijskega prostora so posamezni objekti. V nasprotju z drugimi objektno orientiranimi pristopi metodologija OOSE tako razlikuje tri vrste objektov. Glavni razlog za vpeljavo treh tipov objektov je težnja po izgradnji modela sistema, ki bo čim bolj stabilen in prilagodljiv zahtevam po spremembah. Kriteriji za razdelitev sestavin scenarijev po posameznih tipih objektov:
   - informacije, ki se v sistemu hranijo, so skupaj z naravno povezanim obnašanjem objekta združene v entitetnem objektu;
   - v vmesniških objektih so združene informacije in obnašanje, ki se nanaša na predstavitev sistema okolju;
   - v kontrolnih objektih so združene funkcionalnosti, ki niso naravno povezane s posameznim entitetnim objektom ali pa se nanašajo na več različnih entitetnih objektov.

   Model analize se razlikuje od modela področja sistema. Model področja sistema je precej natančna preslikava realnega sistema, kar za model analize ne moremo trditi. Razlog je v tem, ker želimo postaviti stabilen model, ki bo zagotovil čim enostavnejše vzdrževanje.

   Posledica tega je, da ne preslikamo realnosti v model

[3] *Object-Oriented Software Engineering*

take, kot je, kar navadno priporočajo objektno orientirani pristopi. Model analize sistema odraža realnost na način, kot jo želimo videti z vidika upravljanja in vzdrževanja rešitve.

3. Model načrta sistema. Logični model se z upoštevanjem posebnosti okolja implementacije transformira v model načrta sistema. Ta se zgradi na podlagi modela zahtev in modela analize sistema. Model analize sistema je zgrajen pod predpostavko idealnega okolja implementacije sistema, model načrta sistema pa pri podrobni razgraditvi modela analize upošteva konkretno okolje implementacije sistema.
4. Model implementacije sistema. Na podlagi modela načrta sistema se zgradi model implementacije sistema, ki zajema generiranje in/ali pisanje programskega koda.
5. Model testiranja sistema. Model testiranja sistema se uporablja pri preizkusu ustreznosti posameznih funkcionalnih delov sistema in sistema kot celote. Model zajema dokumentacijo za izvedbo testiranja in analizo rezultatov testiranja.

Zelo pomembne so povezave med posameznimi modeli. Ponavljanje posameznih korakov razvoja sistema in vzdrževanje sistema zahteva ta možnost sledenja posameznim objektom skozi različne modele.

3.2. Sklepne ugotovitve

Metodologija OOSE priporoča izgradnjo različnih modelov v življenjskem ciklu sistema, ki imajo izhodišče v modelu scenarijev. Na koncu si poglejmo lastnosti dobrih modelov:

- so enostavni za uporabo;
- omogočajo lažje razumevanje sistema, ki ga opisujejo;
- omogočajo obvladovanje sprememb v sistemu;
- nudijo dovolj bogat izbor tehnik za opis sistema;
- zagotavljajo ustrezno osnovo za uspešno komuniciranje med udeleženci v razvoju sistema.

4. Primerjalna analiza metodologij OOAD (G. Booch) in OOSE (I. Jacobson)

Objektivnost primerjave med posameznimi metodologijami je zelo vprašljiva, saj so področja primerjave izbrana subjektivno, sama primerjava pa temelji na osebnih izkušnjah, pridobljenih s študijem razpoložljive literature in praktične uporabe posameznih metodologij. Obe predstavljeni metodologiji pokrivata celoten razvojni cikel sistema, primerjava pa se omejuje predvsem na fazo analize sistema.

Posamezne tehnike, ki se uporabljajo pri modeliranju sistema, navadno presojamo glede na stopnjo njihove natančnosti in razumljivosti. Natančno definirana uporaba posamezne tehnike izključuje možnost različnega tolmačenja posameznih rezultatov, razumljivost pa se kaže v hitrem obvladanju posamezne tehnike in enostavni uporabi. Tehnike modeliranja navadno obravnavajo sistem s treh splošnih vidikov (Fowler, 1994, str. 80):

- statične (podatkovne) strukture sistema;
- obnašanja sistema;
- zgradbe sistema.

4.1. Statična struktura sistema

Statična struktura sistema je opisana z razredi, njihovimi svojstvi in povezavami med njimi. Za predstavitev povezav med razredi uporablja metodologija OOAD model povezav med entitetami in svojstvi, metodologija OOSE pa uporablja binarni model povezav. Glavna razlika med modelom povezav med entitetami in svojstvi in binarnim modelom povezav je v načinu predstavitve povezav. V binarnem modelu so predstavljene povezave samo na en način, s povezavo med posameznimi razredi. Model povezav med entitetami in svojstvi pa poleg povezave med posameznimi razredi prikazuje še povezave znotraj razreda z vrstami svojstev (Martin, Odell, 1995, str. 265).

Metodologija OOAD prikazuje svojstva znotraj oznake za razred. Prikazovanje svojstev in operacij znotraj oznak za razrede je zelo nazorno, vendar primerno le za modele sistemov z manjšim številom razredov. Za večje sisteme velja, da je bolj pregledno, če opredelimo svojstva in metode posameznih razredov ločeno od statičnega modela sistema.

4.2. Obnašanje sistema

Za opis obnašanja sistema so v uporabi različne oblike diagramov prehajanje stanj. Sestavljeni so iz posameznih stanj sistema in povezani s puščicami, ki predstavljajo prehode med posameznimi stanji sistema. Diagrami stanja so zelo uporabni za predstavitev majhnih sistemov, pri velikih pa postanejo preveč kompleksni za praktično uporabo. Zato si pomagamo z definiranjem diagramov prehodov stanj za posamezne razrede, ki so dovolj enostavni za razumevanje. Težava pa se pojavi, ko želimo na podlagi množice diagramov za posamezne razrede predstaviti obnašanje celotnega sistema (Graham et al., 1994, str. 214).

Metodologija OOAD priporoča uporabo diagramov prehodov stanj v fazi analize sistema. Diagrame stanj razširi tudi z uvedbo nadstanj in vgnezdenih podstanj. Obnašanje, ki je skupno različnim stanjem, se združi v nadstanje. Z diagrami prehodov stanj je zelo natančno in strogo opisano obnašanje na nivoju posameznega razreda. Obnašanje med razredi na višjem nivoju je predstavljeno manj strogo z diagrami sporočil med razredi, opisi v strukturiranem jeziku in časovnimi diagrami. V metodologiji OOSE so različica časovnih diagramov iz metodologije OOAD diagrami akcij med objekti, kjer se za definiranje obnašanja uporablja strukturiran jezik.

Pristop metodologije OOSE k analizi obnašanja sistema je zelo podoben pristopu, ki ga uporablja metodologija OOAD, vendar se priporoča podrobnejša analiza obnašanja sistema šele v fazi načrtovanja sistema.

4.3. Zgradba sistema

Večje sisteme je zaradi lažjega nadaljnjega razvoja in vzdrževanja sistema koristno razbiti na posamezne podsisteme. Metodologija OOSE uporablja funkcionalni vidik pri razgradnji sistema na posamezne podsisteme, vendar poleg njega upošteva še predviden vpliv sprememb na definirane podsisteme. Cilj analize je torej močna funkcionalna odvisnost znotraj podsistemov in šibka odvisnost med posameznimi podsistemi. Odvisnost med posameznimi podsistemi pomeni, da objekti enega podsistema uporabljajo objekte drugega podsistema. Manjše spremembe v sistemu morajo vplivati le na en podsistem na najnižjem nivoju oz. le na nekatere objekte znotraj tega podsistema. Metodologija OOAD priporoča razgradnjo sistema s pomočjo objektov na višjem nivoju, v katere se združujejo objekti z namenom zagotovitve določenega obnašanja sistema na višjem nivoju. Metodologija OOAD pri objektno orientirani razgradnji sistema tako upošteva komunikacijo in vidljivost med objekti. Objektni diagrami prikazujejo pretok sporočil med objekti, lahko pa jih uporabimo tudi na višjem nivoju za prikaz komunikacij med posameznimi podsistemi. Vidljivost med posameznimi razredi oz. podsistemi pa pomeni možnost, da lahko en razred oz. podsistem izkorišča določene lastnosti drugega.

4.4. Druge primerjave

Metodologija OOAD in še posebej metodologija OOSE posvečata veliko pozornost posameznim scenarijem. Gradnja s pomočjo scenarijev pa lahko pomeni tudi nevarnost, da se sistem zgradi preveč na osnovi tekočih potreb, če se v procesu razvoja informacijskega sistema ne izvede reinženiring poslovnih procesov.

Kriterija za medsebojno primerjavo metodologij sta lahko tudi (Frost, 1994, str. 190-191):

1. Podpora analizi in načrtovanju in lahek prehod med njima. Metodologiji OOAD se očita šibka podpora fazi analize sistema. Metodologija OOSE zagotavlja povezavo med modelom analize sistema in modelom načrta sistema s pomočjo modela scenarijev, ki zagotavlja sledljivost posameznim komponentam skozi različne modele.
2. Zapletenost metodologije. Metodologija OOAD nudi širok izbor tehnik, ki se lahko uporabijo pri izgradnji sistema. Tak izbor po eni strani zagotavlja veliko prilagodljivost, po drugi pa otežuje uporabo, ker ni vedno jasna ustrezna kombinacija postopkov za doseganje dobrega rezultata. Proces razvoja sistema je jasneje določen pri metodologiji OOSE.

Posamezne metodologije lahko primerjamo tudi z vidika uporabnosti (Avison, Fitgerald, 1995, str. 460-462). Za glavni cilj imajo izgradnjo računalnalniško podprtega informacijski sistema, so pa tudi bolj splošno uporabne, npr. za modeliranje procesov v poslovnem sistemu. Metodologija OOSE[4] gre še korak dlje. Z združitvijo z elementi objektno orientiranega inženiringa poslovnih procesov je postala močno orodje za modeliranje poslovnih procesov v poslovnem sistemu (Jacobson I., Ericsson M., Jacobson A., 1995).

5. Zaključek

Na koncu povzemimo glavne značilnosti obravnavanih metodologij:

1. Metodologija OOAD (Booch).
   - Metodologija je sestavljena iz širokega nabora načinov zapisovanja s slabo definiranim procesom izgradnje informacijskega sistema. Navodila razvijalcu sistema v primeru zastoja so zelo skopa.
   - Načini zapisovanja so zelo obsežni in so uporabni za dokumentiranje skoraj vseh vidikov sistema. Razpoložljivi načini zapisovanja lahko v določenih primerih povzročijo zmedo pri dokumentiranju.
   - Metodologija omogoča izgradnjo dobrega objektno orientiranega načrta sistema, podpora fazi analize sistema pa je vprašljiva.
2. Metodologija OOSE (Jacobson).
   - Scenariji zagotavljajo precej pomoči pri izgradnji različnih modelov sistema. Za slabost se lahko izkaže precej skromna grafična podpora.
   - Modeli so preprosti in nazorni, ne predvidevajo pa posebnih prilagoditev v primeru izgradnje velikih sistemov.
   - Model scenarijev predstavlja dobro izhodišče za izgradnjo vseh drugih modelov v celotnem življenjskem ciklu sistema. S tem je zagotovljen sistematičen proces izgradnje sistema.

Literatura

1. AVISON David E., FITZGERALD Guy: Information Systems Development: Methodologies, Techniques and Tools - Second Edition. London: McGraw-Hill International (UK) Limited, 1995, 505 str.

[4] *Metodologija OOSE je pravzaprav povzetek širše metodologije, imenovane Objectory, ki zajema tudi področje inženiringa poslovnih procesov.*

2. BOOCH Grady:
The Booch Method: Process and Pragmatics. Object Development Methods, Edited by Carmichael. New York: SIGS Books, Inc., 1994, str. 149-166.

3. BOOCH Grady:
Object-Oriented Analysis and Design with Applications - Second Edition. Menlo Park, California: Addison-Wesley Publishing Company, 1994a, 589 str.

4. BOOCH Grady:
Object Solutions, Managing the Object-Oriented Project. Menlo Park, California: Addison-Wesley Publishing Company, 1996, 323 str.

5. FOWLER Martin:
Describing and Comparing Object-Oriented Analysis and Design Methods. Object Development Methods, Edited by Carmichael. New York: SIGS Books, Inc., 1994, str. 79-109.

6. FROST Stuart:
The Rumbaugh Method (OMT): The Selection of an Object-Oriented Analysis Method. Object Development Methods, Edited by Carmichael. New York: SIGS Books, Inc., 1994, str. 247-270.

7. JACOBSON Ivar, CHRISTERSON Magnus, CONSTANTIN Larry L.:
The OOSE Method: A Use-Case-Driven Approach. Object Development Methods, Edited by Carmichael. New York: SIGS Books, Inc., 1994, str. 247-270.

8. JACOBSON Ivar et al.:
Object-Oriented Software Engineering, A Use Case Driven Approach. Wokingham: Addison-Wesley Publishing Company, 1995, 528 str.

9. JACOBSON Ivar, ERICSSON Maria, JACOBSON Agneta:
The Object Advantage, Business Process Reengineering with Object Technology. Wokingham: Addison-Wesley Publishing Company, 1995, 347 str.

10. MARTIN James, ODELL James J.:
Object-Oriented Methods: A Foundation. Englewood Cliffs, New Jersey: PTR Prentice Hall, 1995, 412 str.

11. van der WALT Egbert, STEENKAMP Annette L.:
A Revised Spiral Model for Object-Oriented Development. Proceedings of The Fourth International Conference Information Systems Development - ISD'94, Methods & Tools & Theory & Practice. Bled: Moderna organizacija Kranj, 1994, str. 163-172.

◆

Milan Črv je diplomiral na Ekonomski fakulteti v Ljubljani, kjer je opravil tudi magisterij iz informacijsko-upravljalskih ved. Ukvarja se z razvijanjem informacijskih sistemov s področja poslovne in zdravstvene informatike. Sedaj je zaposlen v Kliničnem centru Ljubljana.

◆

Priznanja slovenskim informatikom

Na posvetovanju Dnevi slovenske informatike Portorož '97 je programski odbor posvetovanja podelil:

- tri priznanja za strokovne dosežke na področju informatike
- tri priznanja za prispevke na posvetovanju .

Priznanja za strokovne dosežke na področju informatike so prejeli:

Marin Silič - pri znanje za dosežke pri informatiziranju državnih organov

Stane Štefančič - priznanje za uspešno uvajanje novih metod pri prenovi informacijskih sistemov

Vladislav Rajkovič - priznanje za dolgoletno uspešno delo na področju informatike v izobraževanju

Priznanja za prispevke na posvetovanju je programski odbor podelil na temelju glasovnic udeležencev. Le-ti so ocenili kot najboljše:

Vladimir Batagelj, Andrej Mrvar: Predstavitve obsežnih omrežij

- *kot najzanimivejši referat*

Niko Slavičič, Tomaž Gosar: Neznosna lahkost povezovanja

- *kot najbolje predstavljen referat*

Evelin Vatovec Krmac: Ponovna uporaba - kaj, zakaj, kdaj in kako

- *kot najizvirnejši referat.*