KONFERENCA *OOPSLA 2007* V MONTREALU

Konference OOPSLA (Object-Oriented Programming, Systems, Languages & Applications) prirejajo po različnih krajih v Ameriki. Osmo konferenco OOPSLA[1] je gostil Montreal od 21. do 25. oktobra 2007. Do sedaj se te konference Izumovci še nismo udeležili, čeprav velja za eno izmed najzanimivejših s področja objektnega programiranja. Ključne osebe v tem svetu predstavljajo John McCarthy, Gregor Kiczales, Fred Brooks, David Parnas, Patti Maes in drugi, ki so imeli uvodna predavanja.

V izvirniku je možno vsa predavanja poslušati na spletnem naslovu http://www.oopsla.org/oopsla2007/index.php?page=podcasts/ oziroma http://www.podbean.com/podcast-detail/24402/oopsla-2007/all (prispevki so v formatu pdf, objavljenih pa je tudi nekaj video posnetkov).

V nadaljevanju predstavljam najzanimivejša predavanja.

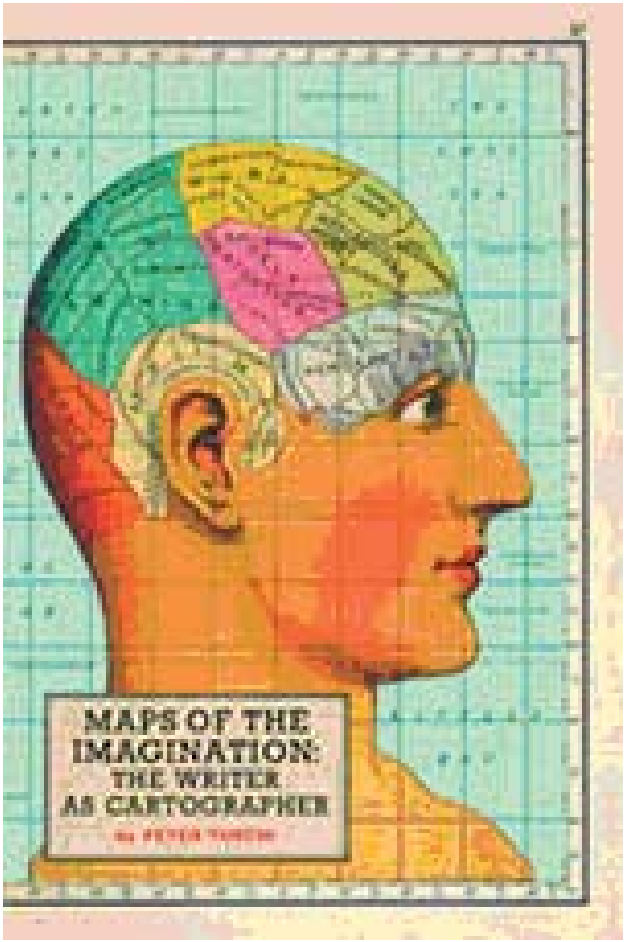
JOHN MCCARTHY: ELEPHANT 2000 – A PROGRAMMING LANGUAGE BASED ON SPEECH ACTS

Dobitnik Turingove nagrade v letu 1971 s področja umetne inteligence je leta 1955 prvi uporabil besedno zvezo umetna inteligenca in spodbudil začetek razvoja te vede, leta 1960 pa je definiral in objavil programski jezik lisp.

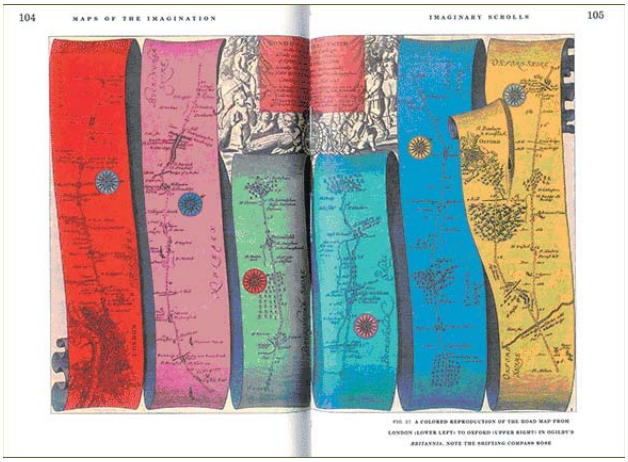
Elephant 2000 je programski jezik za pisanje in preverjanje programov, ki komunicirajo z uporabnikom (tj. procesiranje transakcij) ali s programi v drugih organizacijah (tj. izmenjava podatkov). Komunikacijski vhodi in izhodi se med seboj sporazumevajo v jeziku, katerega stavki so definirani kot vprašanja, odgovori, ponudbe, sprejetje, zavrnitev, povpraševanje, dovoljenje in obljube. Pravilna sintaksa izhaja iz naravnega jezika. Odgovori morajo biti resnični, saj se le tako izpolnijo obljube. Stavke logičnih izrazov je možno generirati iz diagrama, izvorni programi v elephantu pa ne potrebujejo podatkovnih struktur, ker se prepišejo neposredno iz zgodovine. Sam program je po vsem tem času zašel v slepo ulico, saj se nekaterih nepredvidljivih problemov ne da rešiti (na primer potnik kupi karto in jo stornira, pri tem pa program ne ve, ali je potnik lastnik karte ali ne).

PETER TURCHI:[2] IZZIV V PRIPOVEDOVANJU NASLEDNJE ZGODBE

Večina izkušenj, o katerih je avtor govoril na konferenci, je strnjenih v njegovem najbolj znanem delu – knjigi "Maps of the Imagination". Pisanje programa je kakor pisanje zgodbe. Pisatelj se s primeri iz pisanja in kartografije poda na raziskovanje, kako predstaviti odkritje in kako je predstavitev sama ključ k odkritju. Predavatelj je predstavil različne predstavitve in kako te privedejo do popolnoma novih odkritij. Na sliki 1 je predstavil neko poljubno pot po Londonu, kot jo je videl sam in za katero se mu je zdelo, da jo je treba evidentirati (slika 2).



Slika 1: Prva stran knjige Maps of the Imagination



Slika 2: Načrt Londona, kot ga vidi Peter Turchi

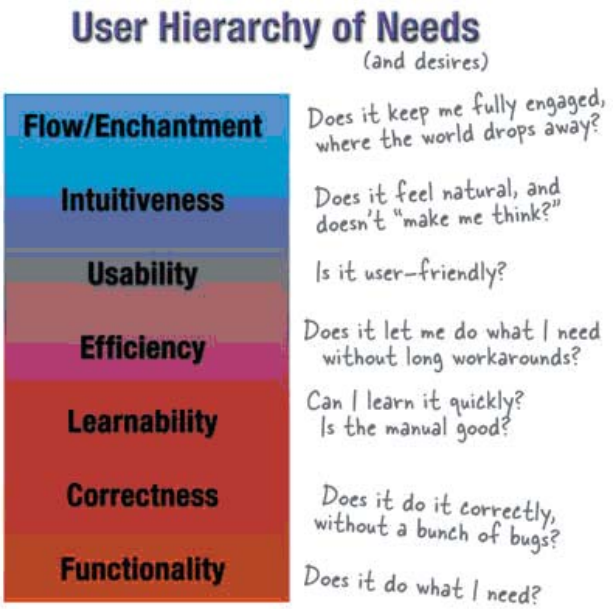
Izkušnje same se ne dajo neposredno pretvoriti v znanje o programiranju, lahko pa programerjevo obzorje razširijo, da na probleme, ki jih rešujejo programi, pogleda z drugega zornega kota. Govoril je o tem, da je dobro, če se tu in tam izgubiš, kajti kadar se izgubiš, moraš najti pot nazaj, v procesu iskanja pa lahko spoznaš novo pot in pridobiš nove izkušnje.

JIM PURBRICK IN MARK LENTCZNER: SECOND LIFE – THE WORLD'S BIGGEST PROGRAMMING ENVIRONMENT

Zelo zanimiv je bil tudi prispevek Jima Purbricka in Marka Lentcznerja, ki kot glavna programerja in ustanovitelja virtualnega okolja z imenom Drugo življenje govorita o prihodnosti skriptnih jezikov. Med predavanjem se je izkazalo, da je ekipa programerjev razkropljena po vsem svetu. Druženje v Drugem življenju je možno v virtualni pisarni na sestankih, za kar si programerji nadenejo podobe. Sedijo eden poleg drugega v skupnem okolju (npr. nekje v ozadju so diagrami poteka, časovni diagrami, programske sheme), pri čemer je možno filtrirati različne glasove po glasnosti in tako slediti pomembnejšemu pogovoru in zanemariti kaj postranskega.

KATHY SIERRA:[3] CREATING PASSIONATE USERS

Če je stvar zanimiva, se kot prvo zmeraj pojavi vprašanje: Kdo se bo lotil dodatnega dela? Kdo bo naredil dodatne vaje in boljša navodila? Odgovor na vprašanje je preprost: Skupnost uporabnikov! Skupnost namreč lahko zagotovi strokovno pomoč, izobraževanje uporabnikov, trženje (od ust do ust), neavtentificirane dodatke kakor tudi nove ideje o izdelku. Obstaja pa tudi možnost dodatnega zaslužka (npr. posebni izdelki, majice, nalepke in drugo).



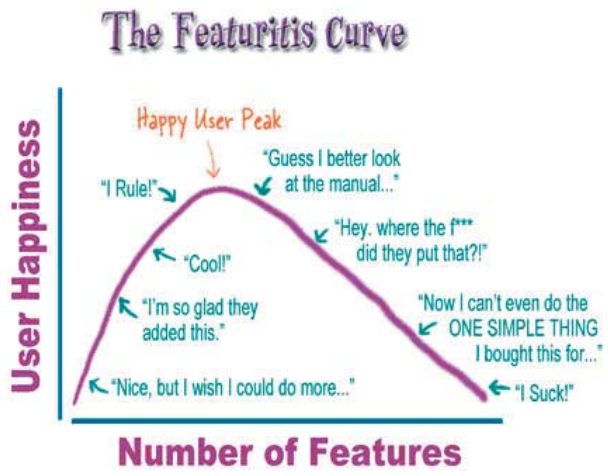
Slika 3: Potrebe uporabnikov

Kako ustvariti skupnost uporabnikov? Treba je:

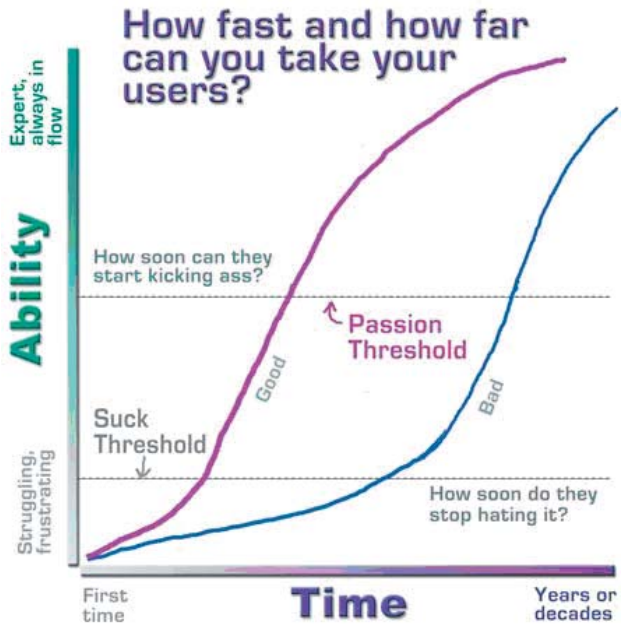
- ustvariti forum za razprave (lahko vsebuje pogovore, wikije in bloge);
- preveriti, ali za naš proizvod že obstaja kak forum in najbolj zagrete uporabnike pridobiti za svoj forum;
- nagraditi zagrete uporabnike s posebnimi privilegiji in priznanji (npr. naziv, dostop do predčasnih izdaj programa, dostop do razvijalcev in ekspertnih skupin – tak primer je skupina javaranch.com s 3/4 milijona mesečnih obiskov);
- upoštevati strokovno podkovane uporabnike, saj lahko s primernimi vprašanji veliko pripomorejo h kakovosti uporabniških priročnikov;
- oblikovati del foruma, kjer lahko uporabniki kramljajo o vsakdanjih stvareh; spodbujati uporabnike, da se srečujejo v realnem svetu (recimo organizacija konferenc);
- ločiti forum na zahtevnostne nivoje, takoj ko se dovolj razširi (preprečevati agresivnost do začetnikov in bolj okornih uporabnikov foruma);
- učiti člane, da učijo sodelavce in druge člane (boljši kot bodo, bolj motivirani bodo za uporabo našega izdelka).

KRATEK POVZETEK ŠE NEKATERIH PREDAVANJ

Torbjörn Ekman in Görel Hedin sta v prispevku *The JastAdd Extensible Java Compiler* predstavila prevajalnik java, ki ga je mogoče razširiti v statično orodje za analizo programov.



Slika 4: Korelacija zadovoljstva uporabnika in funkcionalnosti



Slika 5: Krivulja napredovanja uporabnikov

Andy Georges, Dries Buytaert in Lieven Eeckhout so v *prispevku Statistically Rigorous Java Performance Evaluation* zatrdili, da ni trivialno ocenjevati delovanja jave, saj nanjo vplivajo vhodni podatki, virtualni stroj, odstranjevalec smeti, velikost kopice ... potreben čas za zagon pa se spreminja, tudi kadar so pogoji identični. Pri tem tudi programi za testiranje delujejo različno; eni merijo najboljše čase, povprečje vseh časov, najslabše čase ter se zaganjajo znotraj enega ali več virtualnih strojev. Avtor je poskušal po različnih poteh oceniti delovanje ter ga natančno določiti.

Feng Sian, Witawas Srisa-an in Hong Jiang so v prispevku *MicroPhase: An Approach to Proactively Invoking Garbage Collection for Improved Performance* govorili o tem, da je danes najbolj pogost kriterij za klic smetarja glede na porabo kopice, tj. smetar se kliče, kadar je kopica polna. MicroPhasov princip se razlikuje, saj temelji na preverjanju potrebe po razporeditvi in spremljanju življenjskega cikla objektov. Samo testiranje je potekalo na Sun Microsystems HotSpot-ovem virtualnem stroju, rezultati pa so varirali od 2,5-odstotnega poslabšanja do 14-odstotnega izboljšanja.

Yannis Smaragdakis, Tony Kay, Reimer Behrends, Michal Young so v prispevku *Transactions with Isolation and Cooperation* predstavili, da v modelu transakcij izolacije in kooperacije (TIC) transakcijski spomin omogoča sledenje drugim nitim. V določenih točkah je možno opazovati njihove vrednosti, omogoča pa tudi, da transakcije med seboj sodelujejo: izmenjujejo podatke, kličejo neponovljive ali nepovratne operacije (npr. I/O). Glavna skrb TIC je, da kooperacija med transakcijami ne ogrozi dejanskega stanja kakšne transakcije.

Opombe

1 Podobna konferenca v Evropi je ECOOP (European Conference on Object-Oriented Programming), v Sloveniji pa OTS (Objektna tehnologija Slovenije).

2 Peter Turchi: http://www.peterturchi.com/.

3 Kathy Sierra: http://headrush.typepad.com/.

Andrej Barovič Karpov