Alati za razvoj aplikacija bez kodiranja

Dražen Vukotić, Nikola Tanković
Superius d.o.o., Ćirilometodske družbe 1, 52100 Pula
drazen.vukotic@superius.hr, nikola.tankovic@superius.hr

SAŽETAK

Penetracija računalne opreme među stanovništvom ubrzano se primiče omjeru jedno računalo po korisniku, s tendencijom rasta na više računala po korisniku. Tom trendu značajno pridonose sveprisutniji "pametni telefoni". Kako bi se iskoristio kreativni i potrošački potencijal ogromnog broja korisnika, pojavljuje se potreba da se omogući jednostavno i intuitivno modeliranje, izrada i isporuka vlastitih ili modifikacija postojećih aplikacija kako za stolna, tako i za mobilna računala. Tu potrebu zadovoljava nova generacija razvojnih alata koji omogućavaju izradu aplikacija bez potrebe za poznavanjem specijalističkih znanja o programiranju. U radu se opisuju i klasificiraju takvi alati i tehnike na kojima su zasnovani.

ABSTRACT

Computer hardware is gaining its share in human population towards one computer per user, with a tendency to become more than one computer per single user. Smartphone market is significantly contributing that trend. To use the creative and consumer potential of such vast amount of users, there is a need to enable intuitive and easy modeling, developing and deploying new or modifying existing desktop or mobile applications. This need is satisfying by the new generation of development tools, enabling creating applications with ease and no coding. This work will describe and classify such tools and underlying techniques.

1. Uvod

Svjedoci smo prave poplave raznovrsne računalne opreme u svim segmentima stanovništva. Njihova zajednička karakteristika ogleda se u činjenici da omogućava pokretanje različitih korisničkih programa (aplikacija) koji pokrivaju raznolike potrebe krajnjih korisnika. Na taj trend značajno utječu uređaji koje svrstavamo pod zajednički naziv mobilnog računalstva (*engl. mobile computing*), a u koje ubrajamo *netbook, smartphone i tablet* računala. Njihovoj popularnosti pridonose karakteristike koje klasična stolna (engl. *desktop*) računala nisu imala: mobilnost, dostupnost, intuitivnost korisničkog sučelja korištenjem dodirnog ekrana te ugrađene različite vrste senzora poput kamere, GPS navigacije, raspoznavanja govora ili akcelerometra. Ove karakteristike omogućile su njihovu primjenu u najrazličitijim situacijama i zadacima, od privatnih do poslovnih. Na tržištu se pojavljuje veliki broj aplikacija koje pokušavaju udovoljiti potrebama ogromnog broja potencijalnih korisnika koji se mjeri u stotinama milijuna. Iako je prema istraživanjima Mobile Entertainment-a [Mobile Entertainment 1] 80-90% svih preuzetih (engl. *downloaded*) aplikacija besplatno, a prosječna cijena onih koje se plaćaju je tek 1$, radi se o milijardama preuzimanja i izuzetno značajnom i potentnom tržištu.

Pored ovog trenda, pojavljuje se i trend rasta specijalizirane računalne opreme čijeg postojanja često nismo ni svjesni, a dio su pojave koju nazivamo sveprisutno računarstvo (engl. *Ubiquitous Computing*). Sveprisutno računarstvo naglašava masovnost računala, geografsku raspodijeljenost, uklopljenost u okolinu, umreženost i funkcionalnost. Za razliku od virtualne stvarnosti koja ljude stavlja u svijet računala u kojem je izazov samo računarstvo, sveprisutno računarstvo stavlja računala u svijet ljudi s namjerom poboljšanja kvalitete života. [Žagar M., Projekt:„Programsko inženjerstvo u sveprisutnom računarstvu“ 2]. Rezultat ove pojave je generiranje izuzetno velike količine strukturiranih i nestrukturiranih informacija koje je potrebno na odgovarajući način obraditi i prezentirati korisnicima.

Ova dva trenda dodatno produbljuju jaz između ponude i potražnje kvalitetnih aplikacija koje će zadovoljiti sve veće potrebe korisnika u segmentima kako privatnog, tako i poslovnog života. Zbog toga se sve više radi na iznalaženju novih načina za kreiranje robustnih, skalirajućih, klijentskih ili web aplikacija u što kraćem vremenu i sa što manjom potrebom za specifičnim, specijalističkim znanjima o programiranju.

U nastavku rada opisuje se postupak izrade aplikacija na klasičan način i uz upotrebu alata za razvoj aplikacija bez kodiranja. Nakon toga se daje klasifikacija takvih alata i pregled njihovih tipičnih predstavnika, a na kraju je iznesen zaključak i predviđanje daljnjeg razvoja na ovom području.

2. Klasičan način razvoja aplikacija

Često se pojam razvoja aplikacija i programiranja poistovjećuju iako se radi o bitno različitim procesima. Pojam razvoja aplikacija (engl. *Application Development*) obuhvaća sve faze razvoja računalne aplikacije ili informatičkog sustava koji može biti složen od više komplementarnih aplikacija. Postoje različite metodologije koje se bave definiranjem procesa u razvoju aplikacija. Neke od uobičajenih i općeprihvaćenih faza [McConnell 3] u razvoju aplikacija su: analiza, dizajn, programiranje, testiranje, implementacija, održavanje i podrška.

Ove faze se mogu provoditi kaskadnim („vodopadni“ - engl. *Waterfall*) ili spiralno - iterativnim postupkom. Kaskadni postupak podrazumijeva da je svaka prethodna faza u potpunosti dovršena prije nego se može prijeći na sljedeću fazu. Pogreške u ranijim fazama (analiza i dizajn), teško se ispravljaju u kasnijim fazama i povezane su sa značajnim troškovima ili pomicanjem rokova. Međutim, pogreške ili nesporazumi u fazi analize i dizajna postaju vidljive tek u fazi testiranja što predstavlja velike rizike za projekt. Stoga su razvijeni spiralno – iterativni postupci koji također prolaze kroz slične faze, ali počinju od manjeg broja funkcionalnosti koje se postupno, u iteracijama nadopunjuju novim funkcionalnostima. Cilj spiralno-iterativnih postupaka je što prije dobiti funkcionalni prototip aplikacije i tako smanjiti rizike razvoja. Spiralno-iterativni postupci imaju svoje različite inačice za što brži razvoj, a koje zajednički prema pojmu koji je uveo James Martin [Martin J. 4] nazivamo RAD (*engl. Rapid Application Development*). Neke od najpoznatijih inačica RAD metoda su Agile Software Development, Extreme Programming, Joint Application Design, Lean Software Development i Scrum.

Za razliku od pojma razvoja aplikacija, pojam programiranja predstavlja iako ključan, samo jedan od spomenutih elemenata u procesu razvoja. U nekim slučajevima RAD pristupa, procesi analize, planiranja, dizajna i testiranja implicitno su uključeni u proces programiranja te često dolazi do poistovjećivanja ova dva pojma.

Sam pojam programiranja ili kodiranja označava pisanje programskog koda, odnosno skupa instrukcija računalu koje opisuju način na koji računalo treba izvršiti određeni zadatak [Maćešić N. 5]. Pri tome se koristi odgovarajući programski jezik koji u zavisnosti od stupnja apstrakcije, može pripadati jednoj od 4 generacija programskih jezika: 1GL (binarni kod), 2GL (asemblerski kod), 3GL (viši programski jezici) i 4GL (problemski orjentirani jezici). Nadodavanjem grafičkog sučelja za unos programskih naredbi na jezike treće i četvrte generacije dobija se hibrid kojega proizvođači iz marketinških razloga neopravdano nazivaju 5GL. Svaka sljedeća generacija programskog jezika donosi značajan napredak u produktivnosti programiranja, a u praksi se najčešće prilikom generiranja izvršnog strojnog koda koristi postupak koji prevodi naredbe višeg programskog jezika u niži, sve do 1GL odnosno do strojnog koda.

Tijekom povijesti razvoja informatike, razvijeno je mnoštvo programskih jezika za razne svrhe, od kojih su samo neki postali općeprihvaćeni. Na stranicama HOPL, [The History of Programming Languages 6] katalogizirano je preko 8500 različitih programskih jezika. Razvijene su i posebne metrike za mjerenje zastupljenosti programskih jezika [langpop.com 7], a može se utvrditi da prvih 50 najzastupljenijih jezika koristi preko 95% programera.

Većina aplikacija sastoji se od nekoliko uobičajenih elemenata: ekranskog korisničkog sučelja (engl. *User interface*), programske logike, sustava za pohranu podataka (DBMS) i sustava za izvješćivanje (engl. *Reporting*). U skladu sa time, postoje i posebni programski alati namijenjeni izradi, ispravljanju i testiranju navedenih elemenata, a ponekad su svi zajedno objedinjeni u jedan integrirani razvojni alat - IDE (engl. *Integrated Development Environment*). Ovakvi integrirani razvojni alati ili razvojna okruženja značajno unaprijeđuju i olakšavaju proces programiranja koji po svojoj prirodi zahtijeva popriličnu količinu specifičnog, specijalističkog znanja o korištenom programskom jeziku kao i o uporabi samog razvojnog alata.

Iako postoje alati koji podupiru i ostale faze u razvoju aplikacija, npr. alati za vođenje i upravljanje projektima, alati za planiranje, alati za testiranje, implementaciju i korisničku podršku, sam proces

programiranja je jedini proces koji se može u velikoj mjeri, a ponekad i u potpunosti automatizirati. Moderna IDE okruženja sadrže mnoštvo vizualnih pomagala koja olakšavaju kreiranje ekranskog korisničkog sučelja, modeliranje i povezivanje sa bazom podataka ili izradu kompleksnih izvješća. Međutim, svaka zahtjevnija programska logika koja čini okosnicu aplikacije, mora se napisati u odabranom programskom jeziku. U tom dijelu nastupaju alati za razvoj aplikacija bez programiranja koji su na tragu tome da se postupak programiranja maksimalno olakša, a u nekim slučajevima i ukine.

3. Razvoj aplikacija bez potrebe za kodiranjem

Prilikom razvoja aplikacija redovito se pojavljuje problem nerazumijevanja između osoba koje posjeduju znanje o problemskoj (poslovnoj) domeni za koju se aplikacija izrađuje, osoba zaduženih za analizu predmetne domene i dizajn te osoba zaduženih za izradu aplikacije, odnosno programiranje. Što je veće preklapanje znanja između sudionika u razvoju, to je veća vjerojatnost da će realizirana aplikacija bolje zadovoljavati korisnikove potrebe. U klasičnom pristupu, taj se problem rješava korištenjem određene metodologije (kaskadne, spiralne, iterativne) koje, svaka na svoj način, smanjuju rizik nerazumijevanja. U pristupu razvoja aplikacija bez kodiranja, pokušava se smanjiti ili dokinuti potreba za programerskim ili analitičarskim znanjem i na taj način eliminirati nerazumijevanje, ubrzati razvojni ciklus i olakšati naknadne dorade..

Nakon provedenog istraživanja na gotovo 30 razvojnih alata različitih proizvođača, utvrđeno je da se alati za razvoj aplikacija bez programiranja mogu svrstati u četiri osnovne kategorije:

- Alati zasnovani na principima vizualnog programiranja
- Alati zasnovani na modeliranju
- Alati zasnovani na definiranju domenskih ontologija
- Alati zasnovani na kreiranju novih aplikacija metodom komponiranja elemenata postojećih aplikacija

U nastavku se detaljnije razrađuje svaka od navedenih kategorija.

3.1. Alati za vizualno programiranje

U alate za vizualno programiranje ubrajaju se oni alati koji omogućavaju kreiranje programske logike pomoću slaganja i povezivanja vizualnih elemenata i njihovog prostornog rasporeda umjesto pisanjem tekstualnih naredbi programskog jezika [Wesley, Hanna and Millar 8]. Pri tome se i dalje koriste semantička i sintaksna pravila kao i kod tekstualnih programskih jezika, ali su elementi jezika u ovom slučaju prikazani grafičkim simbolima i piktogramima. Iako govorimo o alatima za razvoj aplikacija bez programiranja, potrebno je naglasiti da je kod ovih alata ipak potrebno poznavati principe programiranja.

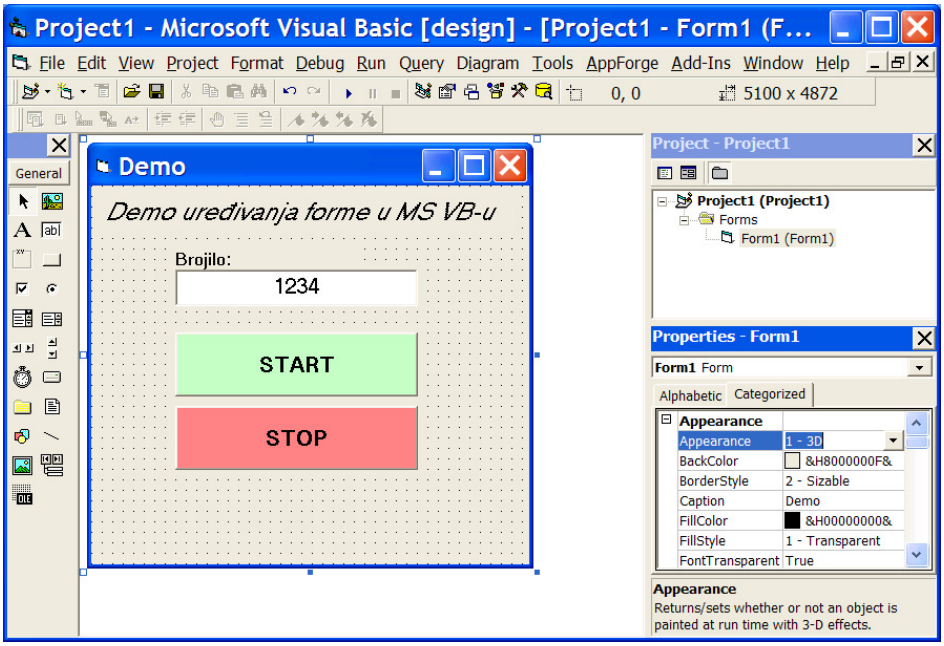
Na tržištu postoji veliki broj alata zasnovanih na 3GL-u ili 4GL-u sa vrlo razvijenim i moćnim dodacima za grafičko uređivanje elemenata korisničkog sučelja na principima povuci-pusti (engl. *drag and drop*) i WYSIWYG (*engl. What You See Is What You Get*) uređivača sučelja. Takve alate ne bi trebalo razmatrati u kategoriji „pravih“ alata za vizualno programiranje jer je njihova baza i dalje klasični programski jezik nadopunjen sa elementima vizualnog programiranja samo u segmentu korisničkog sučelja dok se programska logika koja stoji iza sučelja i dalje definira korištenjem tekstualnih programskih naredbi 3GL programskog jezika. Međutim, zbog kompletnosti prikaza i njihovom značajnom doprinosu na tom području, uvršteni su u posebnu kategoriju.

Kod „pravih“ alata za vizualno programiranje nema potrebe za utipkavanjem programskih naredbi ili drugog teksta, osim u slučajevima kada je radi preglednosti potrebno imenovati elemente programa (varijable, labele, blokove i slično). Provedeno istraživanje je pokazalo da se alati za vizualno programiranje mogu svrstati u tri glavne kategorije:

1. uređenje ekranskih formi i izvješća što je ranije već spomenuto,
2. korištenje dijagrama koji opisuju tijek podataka ili odvijanja procesa,

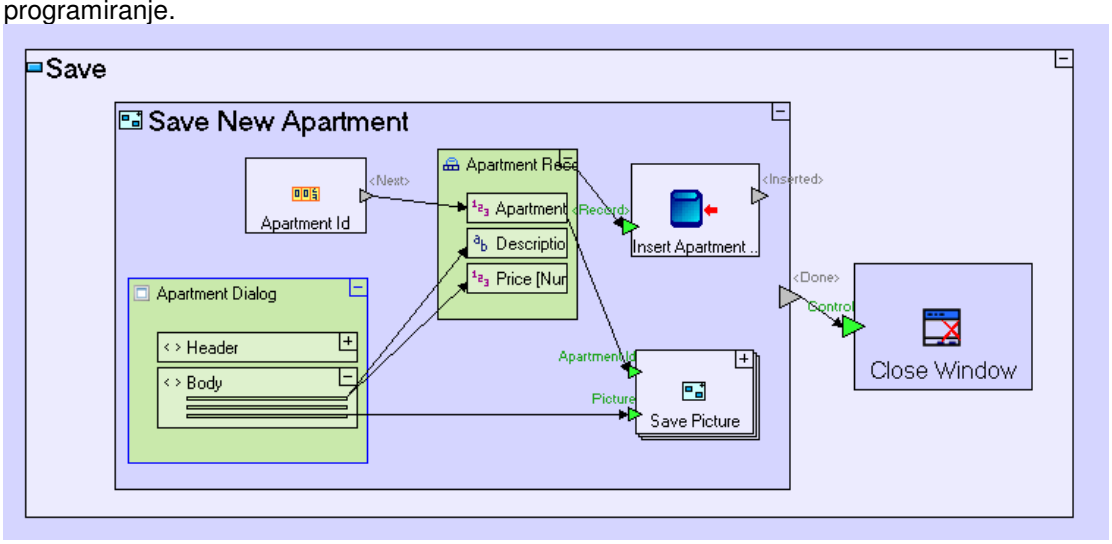
3. grafička reprezentacija programskih logike kao što su pridruživanje, uvjetno grananje ili ponavljanje. Ova kategorija postaje sve popularnija, a trenutačno se najviše koristi kod alata namijenjenih za učenje principa programiranja.

Slika 1 prikazuje izgled tipičnog predstavnika alata koji ima mogućnost grafičkog uređivanja ekranskih formi.



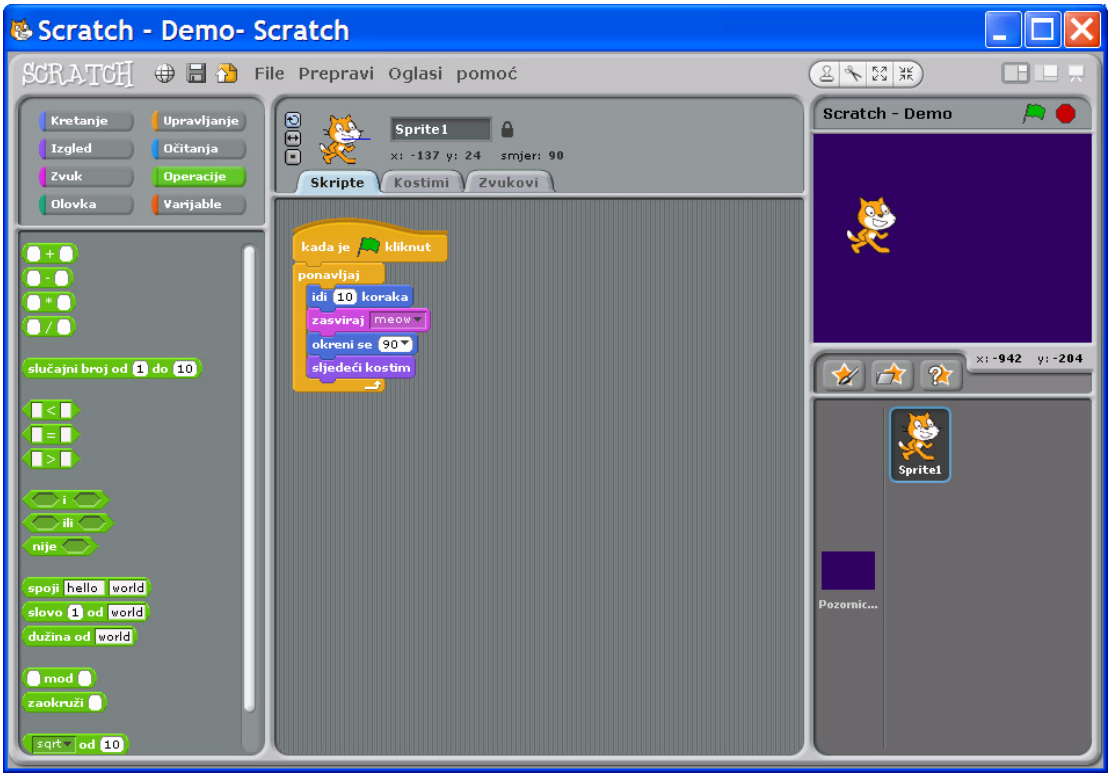
Slika 1: Microsoft Visual Studio - tipičan predstavnik alata koji omogućava grafičko uređivanje ekranskih formi

Slika 2 prikazuje grafičku notaciju poslovnog modela alata koji koristi dijagram toka za vizualno programiranje.



Slika 2: Tersus koji koristi dijagrame toka za vizualno programiranje

Slika 3 prikazuje alat koji omogućava slaganje programske logike potpuno vizualnim postupkom.



Slika 3: Scratch – potpuno grafička reprezentacija programske logike

Budući trend razvoja trebao bi ujediniti osobine sve tri kategorije na način da se u jednom alatu mogu iskoristiti osobine one kategorije koja najbolje odgovara pojedinom poslu pri razvoju nove aplikacije.. Na primjer, za dizajn ekranskih formi ili izvješća najpraktičnija je prva spomenuta kategorija koju prikazuje Slika 1. Za definiranje procesa na višoj razini idealan je dijagram toka (engl. *dataflow*) iz druge kategorije koju prikazuje Slika 2, a za realizaciju svakog pojedinog procesa može se koristiti pristup treće kategorije koji prikazuje Slika 3.

Na opisani način još više bi do izražaja došla osnovna prednost vizualnog programskog jezika koja zahtijeva relativno malo inicijalnog znanja kako bi ga se moglo početi efikasno koristiti. To je zbog toga što su svi elementi programskog jezika, njegove semantike i sintakse vizualno reprezentirani te se korištenjem kontekstne ovisnosti dinamički sužava izbor elemenata koji se mogu upotrijebiti i na taj način korisnika se vodi kroz postupak programiranja.

Svi alati iz ove skupine su u stvari programski alati, a od klasičnih alata za programiranje razlikuju se samo po tome da se programiranje ne provodi utipkavanjem tekstualnih naredbi programskog jezika već slaganjem grafičkih elemenata. Za korištenje ovih alata potrebno je određeno poznavanje principa programiranja i arhitekture programske podrške.

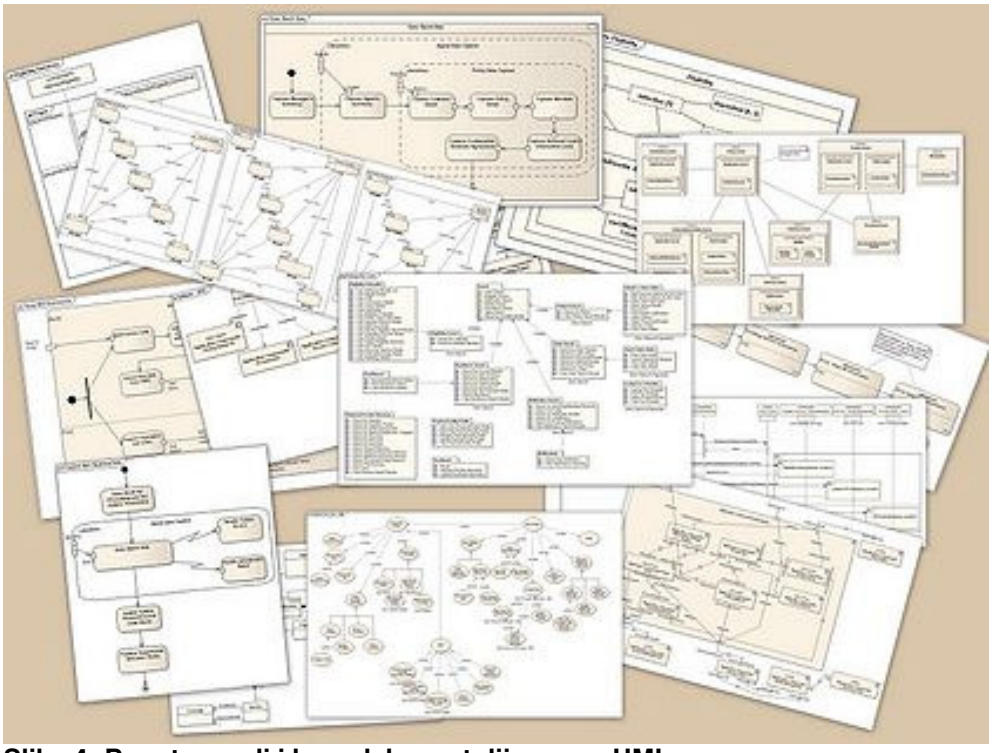
3.2. Alati za modeliranje

Sljedeća skupina alata naglasak stavlja na fazu analize i dizajna, odnosno modeliranje programskog rješenja i pokušava u cijelosti automatizirati proces programiranja i na taj način čim više smanjiti potrebu za njegovim korištenjem u procesu razvoja aplikacija. Ovi alati sadrže više ili manje kompletnije generatore programskog koda za različite 3GL programske jezike (Java, C++). Neki od alata omogućavaju ručnu promjenu izgeneriranog koda koja se može postupkom reverznog inženjeringa vratiti u početni model.

Alati iz ove skupine se mogu dodatno podijeliti u generatore i interpretere. Generatori su oni alati čije rezultat je programski kod koji se prije korištenja mora u cijelosti prevesti (engl. *compile*), što rezultira vrlo dobrim performansama aplikacije prilikom izvršavanja, ali s druge strane nameću i neka ograničenja. Interpreteri najčešće produciraju programski kod na zahtjev što im omogućava izuzetnu fleksibilnost dok su im performanse nešto lošije.

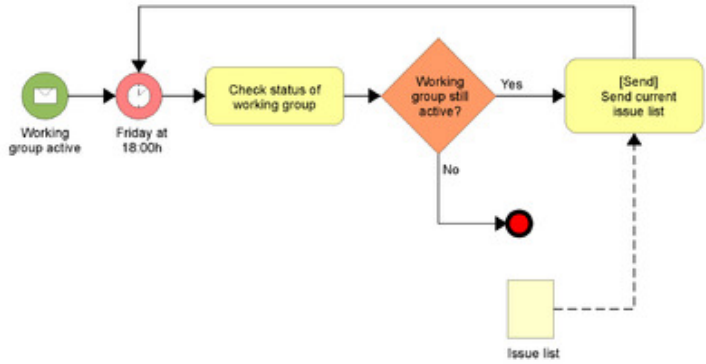
Alati iz ove skupine baziraju se na nekoj od prihvaćenih notacija za analizu, modeliranje i dizajn sustava. Najpoznatiji među njima su UML, BPMN i OPM.

UML (Unified Modeling Language) je standardizirani univerzalni jezik za modeliranje objektno orjentiranih aplikacija [Blaha M., Premerlani W. 9]. Njegova bogata semantika omogućava kreiranje različitih pogleda na isti domenski problem. Međutim, upravo zbog svoje kompleksnosti koja se tijekom vremena sve više povećavala, postao je nepraktičan za korištenje od strane neinformatičkih korisnika. Zbog toga sve teže može ispuniti svoju osnovnu zamisao kao sredstvo nedvosmislene komunikacije između različitih sudionika u procesu razvoja aplikacije. Slika 4 prikazuje mnoštvo različitih tipova UML dijagrama koji omogućuju modeliranje iz više aspekata sustava, no znatno pridonose složenosti samog modeliranja.



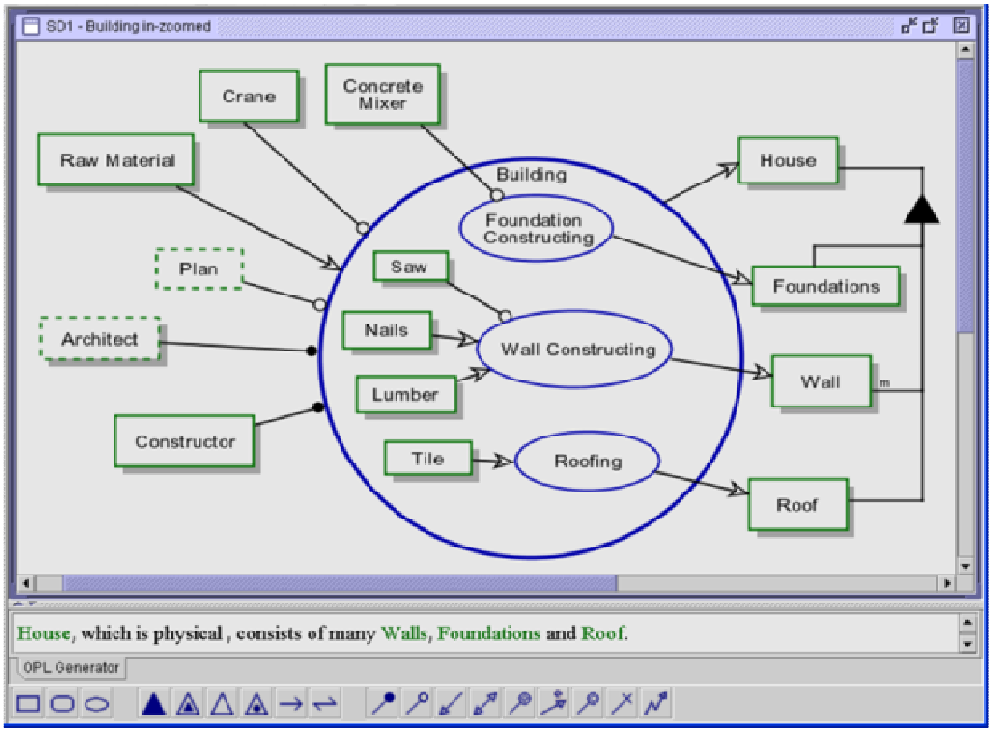
Slika 4: Bogatsvo, ali i kompleksnost dijagrama UML-a

BPMN [*Business Process Modeling Notation* 10] predstavlja grafičku notaciju za prikaz modela poslovnih procesa. Vrlo je slična dijagramima aktivnosti UML-a, a zbog svoje jednostavnosti bolje od UML-a ispunjava zadaću da na jedinstven i intuitivan način omogući opisivanje poslovih procesa. Ali, upravo zbog svoje jednostavnosti, alati koji osiguravaju generiranje aplikacija temeljenih na BPMN-u nemaju onu razinu detaljnosti koja se može postići UML opisom. Slika 5 prikazuje jednostavan proces opisan BPMN notacijom.



Slika 5: Prikaz jednostavnog procesa u BPMN notaciji

OPM (engl. *Object-Process Methodology*) je pokušaj da se ostvari metodologija koja će imati jednostavnost prikaza poput BPMN-a, ali i dovoljnu razinu detalja poput UML-a. Dok UML podržava različite poglede na isti sustav, OPM omogućava putem jedne vrste grafičke reprezentacije prikazati strukturalne, funkcionalne, procesne i arhitekturalne aspekte kroz jedinstven grafički model [Renhartz-Berger, Dori 11]. Na taj način omogućena je izrada refleksivnog, balansiranog, sveobuhvatnog modela iz kojega je puno jednostavnije izgenerirati, a kasnije i održavati programski kod. Zbog svoje kompaktne prirode, OPM ima jedinstveno svojstvo da se model može izražavati kako grafičkim (OPD – Object Process Diagram), tako i tekstualnim prikazom (OPL – Object Process Language). OPL je kontrolirani jezik sličan govornom jeziku te ga je vrlo lako čitati i pisati, a potpuno je svejedno da li se model unosi kroz OPD ili OPL. Slika 6 prikazuje poslovni proces u građevinarstvu u OPG i OPL notaciji (OPL rečenica glasi: *„House, which is physical, consists of many Walls, Foundations and Roof*").



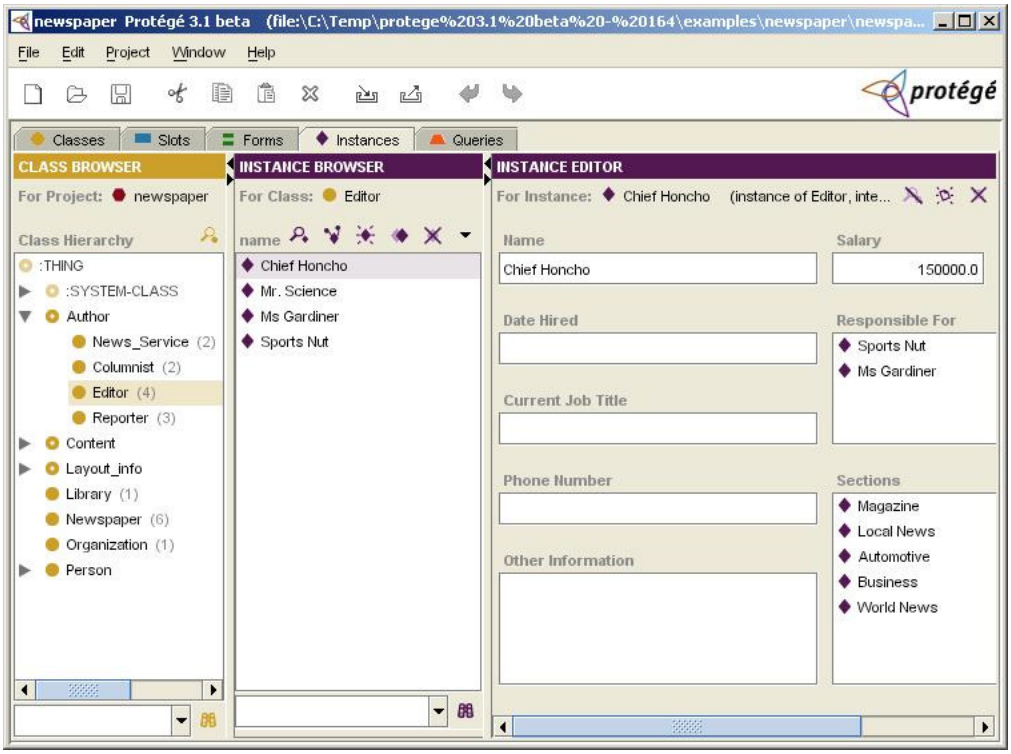
Slika 6: Prikaz OPM modela poslovnog procesa (OPCAT)

3.3. Alati za definiranje ontologija

U posebnu skupinu alata za razvoj aplikacija bez programiranja spadaju alati koji se temelje na izradi ontološke definicije problemske domene. Ontologija u informatičkoj terminologiji predstavlja model

znanja koje opisuje promatrano područje (domenu) kao skup koncepata i veza među njima. Mogu se definirati dvije temeljne grupe ontologija: područna (engl. *domain-specific ontology*) i temeljna (engl. *foundation ili upper ontology*). Područna ontologija opisuje koncepte iz različitih specifičnih domena, a temeljna ontologija opisuje zajedničke koncepte.

Alati za razvoj aplikacija temeljeni na definiranju ontologija omogućavaju brzi razvoj aplikacije koja će omogućiti definiciju entiteta i njihovih veza sa svojim deklarativnim pravilima (klase, atributi i veze) te korištenje automatski prilagođenih ekranskih formi za unos ontologijom definiranih podataka (instanci). Međutim, zahtjevniji procesi mogu se realizirati tek korištenjem ekstenzija napisanih u klasičnom programskom jeziku.
Slika **7** prikazuje korisničko sučelje jednog od najpopularnijih alata za definiranje ontologija.



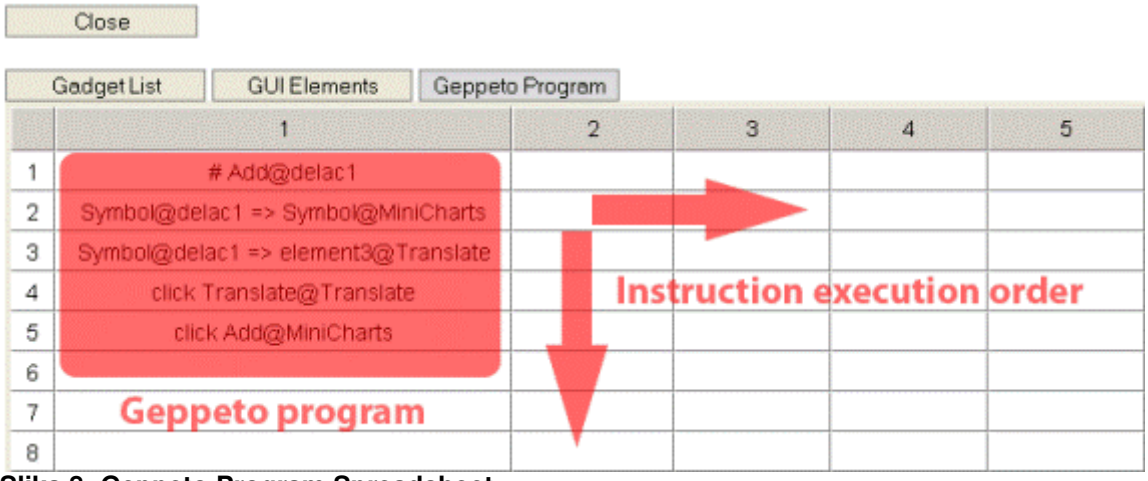
Slika 7: Protege Frame Editor

3.4. Alati za komponiranje

Alati za komponiranje zasnovani su na potpuno drugačijem pristupu razvoju aplikacija. Oni se baziraju na dvije ključne činjenice:

1. Svaka aplikacija namijenjena korisniku, mora sve svoje funkcionalnosti eksponirati putem ekranskog korisničkog sučelja bez obzira radi li se o grafičkom ili komandno-linijskom sučelju.
2. Postoji (ili ga profesionalci mogu izraditi) dovoljan broj elementarnih aplikacija koje rješavaju određeni zadatak koji je potreban korisniku.

Ukoliko su ispunjena oba gornja preduvjeta, pretpostavka je da si korisnik može složiti (komponirati) po volji kompleksnu aplikaciju koristeći gotove aplikacije kao građevne blokove. Ovi alati rade tako da ih korisnik u postupku inicijalizacije „nauči“ kako treba izvesti određenu operaciju koristeći jednu ili više aplikacija. Nakon toga, alat za komponiranje ponavlja „naučeni“ postupak sa zadanim parametrima na isti način kao da je aplikaciju korisnik upotrebljavao ručno. Neki od ovih alata imaju mogućnost primjenjivati i kompleksnu logiku koja će osigurati uvjetno grananje ili ponavljanje. Slika 8 prikazuje Geppeto Program Spreadsheet u kojem je moguće mijenjati redoslijed ili definirati paralelne aktivnosti komponirane aplikacije.



Slika 8: Geppeto Program Spreadsheet

4. Analiza alata za izradu aplikacija bez kodiranja korištenih u istraživanju

U ovom poglavlju dan je tablični prikaz analiziranih alata koji u manjoj ili većoj mjeri odgovaraju specifikaciji alata za razvoj aplikacija bez programiranja. Alati su svrstani u prethodno definirane i opisane kategorije i dan je njihov komercijalni naziv, proizvođač, web adresa i originalni proizvođački opis alata.

4.1. Alati za vizualno programiranje

Naziv	Proizvo-đač	URL	Proizvođački opis
Wawemaker	VMware	http://www.wavemaker.com/	WaveMaker is the only open and easy-to-use development platform for web and cloud applications. With WaveMaker's visual, drag and drop tools, any developer can start building enterprise Java applications with minimal training. WaveMaker creates standard Java applications, boosting developer productivity and quality without compromising flexibility.WaveMaker applications are cloud-ready and include built-in support for multi-tenancy and elastic scaling.
Caspio	Caspio Inc.	http://www.caspio.com/	The Caspio Bridge online database platform is a proven productivity tool for business and IT professionals to create web forms, database searches andweb applications fast and without programming. Cloud computing has never been easier or more cost-effective. Caspio provides everything you need to create and deploy your own web applications in a secure web-based account.
Iceberg	Iceberg Inc.	http://www.geticeberg.com/	Iceberg is a complete workflow software platform driven by BPM on .NET. Drag and drop to create your database, forms and layout. Visually design workflow and process to drive it. Customize the layout to your exact design
Force.com	Salesforce.com Inc.	http://www.salesforce.com	The leading cloud platform for business apps. Every business needs apps: HR apps, inventory apps, iPhone, iPad, Android, and BlackBerry apps. Now you can use the Force.com platform to build all of your apps and websites quickly and easily. 100% cloud requires no hardware or software. Mobile run your apps on any platform or device. Social add collaboration features to every app.

Omnibuilder	OmniSphere information systems corp	http://www.omnibuilder.com	OmniBuilder is a tool which manages the full lifecycle of the application. OmniBuilder generates complete, fully functional applications in a variety of technologies. It captures the full business model, including business rules, requirements and Use Cases. Business templates can be imported to jump-start the application.Real-time prototypes are driven from the business model in an iterative approach to application building. The fully functional prototypes present and manage objects, relationships, triggers, business rules, master-detail screens, windows, menus, security, etc.
Ironspeed	Iron Speed, Inc	http://www.ironspeed.com	Simply point to an existing database and let Iron Speed Designer create a visually stunning, feature-rich Web 2.0 application that is easy to customize and ready to deploy. In just minutes you'll get a complete .NET Web application without programming. Applications are a snap to customize using our simple layout editor, wizards and toolbox. No knowledge of HTML or .NET is required!
Tersus	Tersus Software Ltd.	http://www.tersus.com/	100% Visual - No Coding, No Scripting. Tersus is a Visual Programming Platform for creating rich web and mobile applications. Simply draw flow diagrams and Tersus will bring your application to life.Tersus is open source.
MetaCASE	MetaCase Inc.	http://www.metacase.com/	MetaEdit+ enables companies to radically improve development productivity and quality by generating full code directly from models. First you design the modeling language with MetaEdit+ Workbench and then other developers model with the language in MetaEdit+ Modeler.
Appinventor	Google	http://appinventor.googlelabs.com	To use App Inventor, you do not need to be a professional developer. This is because instead of writing code, you visually design the way the app looks and use blocks to specify the app's behavior. The App Inventor team has created blocks for just about everything you can do with an Android phone, as well as blocks for doing "programming-like" stuff-- blocks to store information, blocks for repeating actions, and blocks to perform actions under certain conditions. There are even blocks to talk to services like Twitter.
Limnor	Longflow	http://www.limnor.com/	Limonor Codeless Programming System is a unique solution that allows users to create their own fully functional applications without writing a single line of code! This development platform uses ready components called Performers and enables the developer to specify their properties, define reactions to system events and link them with other components. The whole process is akin to building a Lego house, but in this case, you will create applications that can actually be used in your daily work!
Scratch	Scratch je razvijen u Lifelong Kindergarten grupi na MIT	http://scratch.mit.edu/	Scratch je programski jezik koji vam na jednostavan i pristupačan način omogućava stvaranje vlastititih interaktivnih priča, animacija, igara, glazbe i umjetnosti, a koje kasnije možete oglasiti i podijeliti na webu s drugima. Dok djeca stvaraju i dijele Scratch projekte, oni istodobno usvajaju i bitne matematičke i računarske koncepte, a istovremeno i uče razmišljati na kreativniji i sistematičniji način, često uz poticajan timski rad.
Alice	The Alice Project is a multi-university initiative, and the Alice Team is a collaboration among faculty, staff	http://www.alice.org/	Alice is designed solely to teach programming theory without the complex semantics of production languages such as C++. Users can program with arrow keys and other controls. Alice can be used for 3D user interfaces.Alice is conjoined with its IDE. There is no syntax to remember. However, it supports the full object-oriented, event driven model of programming.Alice is designed to appeal to specific subpopulations not normally exposed to computer programming, such as female students of middle school age, by encouraging storytelling, unlike most other programming languages which are designed for computation.

4.2. Alati za modeliranje

Naziv	Proizvo-đač	URL	Proizvođački opis
OPCAT studio	Opcat	http://www.opcat.com	OPCAT Studio incorporates the following features: Single model that enables clear and concise expression of hardware, software,humans and regulation. Complexity management by refinement/abstraction mechanisms that include in-zooming/out-zooming, unfolding/folding and suppression/expression. Animated system and product simulation for design-level conceptual debugging and effective early error avoidance or trapping, resulting in huge tangible time and money savings down the road. Automated documentation, code generation and automatic document referencing mechanism. Import and validation functions for system evolution that follows configurable organizational development policies.
Rational Rose	IBM	http://www-01.ibm.com/software/rational/	**I**BM® Rational Rose® Data Modeler offers a sophisticated visual modeling environment for database application development**.** Accelerate your processes by connecting the database designers to the rest of the development team through a common tool and the Unified Modeling Language™ (UML™) v1.4
Rhapsody	Ilogix	http://dsp-fpga.com/i-logix-rhapsody	Rhapsody is the industry's leading Model-Driven Development (MDD) environment for systems, software, and test. A seamless environment for Systems and Software Development with complete design portability. Flexible design environments supporting SysML, UML 2.0, DoDAF, and Domain Specific Languages. Advance auto documentation generation. Design for Testability which includes: Model simulation Requirements Based testing Auto test generation Embedded target model level debugging. Complete application generation for C, C++, Java, Ada Rules-based code generation COM/CORBA generation Code visualization and reverse engineering
PointDragon	GraphLogic	http://www.pointdragon.com/	Pointdragon™ is a web-based environment for developing and running internet applications. pointdragon™ is designed for both the casual end-user and the professional software developer. Anyone with an idea and access to the web can build, use, share, and sell software applications. Welcome to the power and ease of pointclick – dragdrop visual software development.

4.3. Alati za definiranje ontologija

Naziv	Proizvo-đač	URL	Proizvođački opis

Protégé	Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine.	http://protege.stanford.edu	Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. Further, Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications.

4.4. Alati za komponiranje

Naziv	Proizvođač	URL	Opis
Geppeto	Projekt FER-a sponzoriran od strane	http://www.geppeto.fer.hr	Geppeto (Gadget Parallel Programming Tool) is a consumer-level programming environment designed for gadget composition. In this case, the building blocks used to create an application are gadgets. The Geppeto environment itself is a gadget designed to run on the iGoogle page.
Sikuli	Open-source research project developed	http://sikuli.org/	Sikuli is a visual technology to automate and test graphical user interfaces (GUI) using images (screenshots). Sikuli includes Sikuli Script, a visual scripting API for Jython, and Sikuli IDE, an integrated development environment for writing visual scripts with screenshots easily. Sikuli Script automates anything you see on the screen without internal API's support.

5. Zaključak

Kao rezultat provedenog istraživanja, uočena su tri dugoročna trenda u IT sektoru: sve veća količina računalne opreme, sve veći broj njihovih korisnika i sve veća količina generiranih podataka. Ti trendovi dovode do sve veće potrebe za raznovrsnim aplikacijama koje će moći na odgovarajući način pohraniti, obraditi i prezentirati generirane podatke širokoj bazi korisnika na različitim računalnim platformama. Klasičan način razvoja aplikacija programiranjem, teško može udovoljiti ovim potrebama zbog nedostatka stručnog programerskog kadra, dugog razvojnog ciklusa i složenog procesa razvoja i održavanja. Kao odgovor na ove probleme, sve više se razvijaju alati za razvoj aplikacija bez programiranja koji imaju potencijal značajno pojednostavniti i skratiti razvojni proces i omogućiti izradu korisničkih aplikacija svakom zainteresiranom korisniku bez potrebe za specifičnim specijalističkim znanjima iz programiranja.

Napravljeno je istraživanje tržišta alata i metodologija za razvoj aplikacija bez programiranja i temeljem tog istraživanja kreirana je jedinstvena podjela na sljedeće kategorije:

1. Alati za vizualno ili grafičko programiranje
2. Alati za modeliranje
3. Alati za definiranje ontologija
4. Alati za komponiranje

Nakon provedene analize trendova na ovom području i tipičnih predstavnika svake pojedine kategorije, može se zaključiti kako je trend razvoja aplikacija bez potrebe za programiranjem sve izraženiji i kako se u budućnosti očekuje uzlet novih ili značajne nadogradnje postojećih alata, koji će postati općeprihvaćeni od velike baze korisnika. Takvi alati trebali bi biti zasnovani na najboljim praksama danas popularnih, zrelih razvojnih alata i okruženja u području kreiranja ekranskog korisničkog sučelja i generiranja izvješća, zatim grafičkog modeliranja i vizualnog programiranja. Veliki doprinos očekuje se od skupine alata zasnovanih na definiranju ontologija kao i na principima komponiranja. Poseban naglasak alati će morati staviti na iskorištavanje senzora ugrađenih u mobilne,

komunikacijske i specijalizirane uređaje te dodirnog ekrana i upravljanu govorom ili gestama. Osim toga, aplikacije će morati biti skalabilne, multiplatformne i podržavati tzv. multitenant rad. Oni alati koji na jednostavan i intuitivan način budu ponudili veći broj navedenih kriterija, imaju veliku šansu da postanu standard na ovom području.

Literatura

[1] Mobile Entertainment, 157 App Stats you should know about, http://www.mobile-ent.biz/news/read/157-app-stats-you-should-know-about, pregledano 23.05.2011.
[2] Žagar M. Projekt: Programsko inženjerstvo u sveprisutnom računarstvu Ministarstva znanosti, obrazovanja i športa Republike Hrvatske
[3] McConnell, Steve. "7: Lifecycle Planning". Rapid Development. Redmond, Washington: Microsoft Press. pp. 140.
[4] Martin J., RAD, Rapid Application Development, MacMillan Publishing Co., New York, 1990.
[5] Maćešić N., Leksikon računarskih pojmova, VPA Zagreb, 1986.
[6] HOPL, the History of Programming Languages, http://hopl.murdoch.edu.au/, pregledano 23.05.2011.
[7] langpop.com, Programming Language Popularity, http://langpop.com/, pregledano 23.05.2011.
[8] Wesley M. Johnston, J. R. Paul Hanna, and Richard J. Millar, Advances in Dataflow Programming Languages, University of Ulster
[9] Blaha M., Premerlani W., Object-Oriented Modeling and Design for Database Applications, 1998 by Prentice-Hall Inc. ISBN 0-13-123829-9
[10] Object Management Group/Business Process Management Initiative, dostupno na http://www.bpmn.org/, pregledano 24.05.2011.
[11] Reinhartz-Berger I., Dori D., Object-Process Methodology (OPM) vs. UML: A Code Generation Perspective

Podaci o autorima

Dražen Vukotić (drazen.vukotic@superius.hr) diplomirao je na ETF-u Zagreb 1990. godine. U preko 20 godina profesionalnog rada u informatici, radio je na poslovima od sistem analitičara do direktora za tehnologiju i razvoj. Rukovodio je dizajnom, izradom i implementacijom složenih, integralnih informacijskih sustava iz domene proizvodnje, javnog zdravstva, osiguranja, turizma i prodaje. Profesionalna preokupacija su mu izrada univerzalnih, multiplatformskih, multidomenskih računalnih programa za razvoj aplikacija bez kodiranja. Osnivač je i suvlasnik IT tvrtke Superius. Trenutačno pohađa poslijediplomski doktorski studij na FER-u Zagreb.

Dražen Vukotić (drazen.vukotic@superius.hr) graduated in 1990. at ETF Zagreb. In over 20 years of professional experience in IT, he worked on the positions from the System Analyst to the R&D director. He managed design, development and implementation of complex, integrated information systems in manufacturing, public health, insurance, tourism and commerce domain. His professional focus is creating universal, multiplatform, multidomain software for codeless developing. He is founder and co owner of Superius. Currently, he is a PhD student at the FER Zagreb.

Nikola Tanković (nikola.tankovic@superius.hr) je stekao diplomu računarstva na Fakultetu elektrotehnike i računarstva u Zagrebu. Trenutno stječe titulu doktora znanosti na istom fakultetu radeći na području modeliranja aplikacija. Također drži poziciju voditelja razvojnog tima u tvrtci Superius iz Pule. Njegova područja interesa uključuju izvodive aplikacijske modele, graf baze podataka i dizajn web aplikacija.

Nikola Tanković (nikola.tankovic@superius.hr) graduated with Master's Degree in Computer Science at Faculty of Electrical Engineering and Computing in Zagreb. He is currently a PhD student at same university studying in the field of application modelling. He also holds a Development Team Leader position at Superius Ltd in Pula, Croatia. His fields of interest include executable application models, graph databases and web application design.