Razvoj softverskog rešenja za podršku upravljanju proizvodnim nalozima u industrijskoj proizvodnji

Bojan Stojković, Darko Stefanović, Dejan Rašić, Goran Pilipović, Slavica Mitrović
Departman za Industrijsko inženjerstvo i menadžment
Fakultet tehničkih nauka, Univerzitet u Novom Sadu
Novi Sad, Srbija
bstojkovic.88@gmail.com, darkoste@uns.ac.rs, rasomon@uns.ac.rs, goran.pilipovic@telventdms.com, mslavica@uns.ac.rs

Sadržaj- **Prava vrednost aplikacija koje služe kao podrška informacionom sistemu preduzeća ogleda se u dve glavne stvari: prvo treba da obezbedi nesmetano funkcionisanje ne bi li krajnjim korisnicima omogućila da efektivno i lako izvršavaju svoje zadatke, a drugo je da obezbedi pristup značajnim izveštajima i nalozima koji će da pomognu menadžmentu da donosi bolje poslovne odluke. Uspešnost proizvodno-poslovnog sistema u savremenim uslovima poslovanja je između ostalog diktirana načinom i brzinom reagovanja sistema na promene tržišta. Poslovni procesi moraju biti realizovani u najkraćem mogućem roku a utrošak resursa mora biti sveden na minimum. U radu je prikazan postupak razvoja i projektovanja funkcionalnog prototipa za podršku upravljanju proizvodnim nalozima u preduzećima u Oracle razvojnom okruženju koje obuhvata modelovanje funkcionalne strukture, konceptualne i implementacione šeme baze podataka kao i razvoj ekranskih formi i izveštaja.**

Ključne riječi: Oracle; Informacioni sistemi; Proizvodni nalozi; Jdeveloper

I. UVOD

Softversko inženjerstvo je prošlo dug put od kada je kao pojam prvi put upotrebljeno 1968. godine. Pre desetak godina samo su malobrojni mogli da predvide na koje će sve načine softerski proizvodi ući u živote ljudi. Zbog toga je čvrsto teorijsko i praktično utemeljenje softverskog inženjerstva neophodno za razumevanje izrade dobrog i kvalitetnog softverskog proizvoda. Izrada dobrog softverskog proizvoda sadrži elemente umetnosti koja je oličena u razumevanju načina uopštavanja i modelovanja suštinskih elemenata problema, a zatim korišćenju takvih apstrakcija za projektovanje rešenja [1]. U današnjem vremenu se nameće potreba što bržeg i efikasnijeg obavljanja poslova, tu se pre svega misli na to da rešavanje istih poslova zastarelim postupcima i metodama polako postaje relikt prošlosti, zbog sve većeg progresa nauke na mnogim poljima pa i kada je u pitanju obrada podataka, vođenje evidencija i sl. Postoji nekoliko načina prevazilaženja datog problema ali kao najefikasnije nameće se uvođenje automatizovanog informacionog sistema preduzeća u cilju prevazilaženja postojećih prepreka.

Razvoj kompleksne aplikacije je često komplikovan proces koji može da uključuje velik broj razvojnih inženjera sa različitim znanjima i iskustvima koji treba da rade zajedno kao tim u cilju završetka aplikacije unutar postavljenog vremenskog okvira a u isto vreme ta aplikacija mora da bude dovoljno kvalitetna. U početku je softverski proizvod bio sinonim za program ali ovo je promenjeno nakon pojave polja softverskog inženjerstva. Softverski proizvod se više nije posmatrao kao program već kao kombinacija dokumenata i programa.

Softverski proizvodi imaju svoj životni vek koji čini nekoliko faza. U svakoj od faza dolazi do razvoja dela aplikacije koji je značajan za celokupan sistem. Tipični životni vek softverskog proizvoda može se opisati dobro poznatim modelom “Vodopada” [2].

U stvarnosti sam životni vek softvera je mnogo složeniji nego što je to prikazano ovim modelom, ali ipak ovaj model prikazuje osnovne faze koje postoje u svakom procesu razvoju softverskog proizvoda. Model se sastoji iz pet faza : zahtevi – specifikacije - analiza; dizajn; implementacija; testiranje; isporuka i održavanje.

Na osnovu sastavnica i tehnoloških karti, operativnog plana proizvodnje i planova kapaciteta sistema proces pripreme procesa rada ima zadatak da izvrši proveru stanja pripremljenosti resursa i oblikuje proizvodne naloge za otpočinjanje izvođenja procesa rada u sistemu. Provera stanja pripremljenosti resursa je neophodna za kontinuirano odvijanje proizvodnje, a oblikovanje proizvodnih naloga ima za cilj da pruži učesnicima na radnim mestima nosioce informacija u cilju promene stanja ulaznih veličina, tj. podloge za konačno oblikovanje proizvoda. Proizvodni nalog obuhvata radni nalog, radnu listu i trebovanje koji zajedno predstavljaju osnovne nosioce informacija za izvođenje procesa rada i kontrolu tokova [3].

Cilj ovog rada predstavlja razvoj softverskog proizvoda za podršku informacionom sistemu preduzeća za automatizaciju funkcije upravljanja proizvodnim nalozima u Oracle razvojnom okruženju. Struktura rada je sledeća: Metodologija rada, Konceptualno i Implementaciono projektovanje baze podataka

i na kraju Razvoj programske podrške i prikaz prototipa aplikacije za podršku upravljanju proizvodnim nalozima.

II. METODOLOGIJA RADA

Prilikom razvoja predmetnog informacionog sistema korišćen je objektno orijentisani razvoj, pre svega zbog niza prednosti koje sa sobom nosi a tu se pre svega misli na to da se ova metodologija oslanja na objektno orijentisanu paradigmu.

Objektno orijentisani razvoj softvera sastoji se od tri glavna procesa:

- Objektno - orijetisana analiza,
- Objektno - orijentisani dizajn i
- Objektno - orijentisana implementacija.

Objektno orijentisano programiranje predstavlja paradigmu u okviru koje se koncepti posmatraju kao objekti, gde svaki od tih objekata ima identitet, ima svoje stanje, i ima svoje ponašanje. Pored objekata postoje još i procedure koje se u ovoj paradigm nazivaju metode. [4].

A. *Java*

Java je programski jezik razvijen od strane James Goslinga, Sun Microsystems, i objavljen je 1995. godine. Iako sintaksa liči na sintaksu C ili C++ ima jednostavniji model objekta. Jedna od glavnih karakteristika Jave jeste portabilnost, što znači da računarski program napisan u Java programskom jeziku funkcionišu na isti način na bilo kojoj programskoj platformi ili konfiguraciji. Ovo je omogućeno zbog toga što se Java kod kompajlira u Java byte kod umesto da se pretvori direktno u mašinski kod, samim tim može da se izvršava na bilo kojoj virtuelnoj mašini bez obzira na operativni sistem. Takođe Java upravlja memorijom automatski bez pokazivača ili adresiranja.

Java podržava tri glavna principa objektno orijentisanog programiranja:

- **Nasleđivanje**
- **Enkapsulacija**
- **Polimorfizam**

Java EE definiše komponente – tehnologije kao što su *Java Server Pages* kod koji se dalje spaja sa drugim komponentama da bi kreirali aplikaciju. Java EE arhitekturalni model razdvaja ove komponente u logičke slojeve: *Client tier, Web tier, Business tier, EIS tier* [5].

B. *JDeveloper*

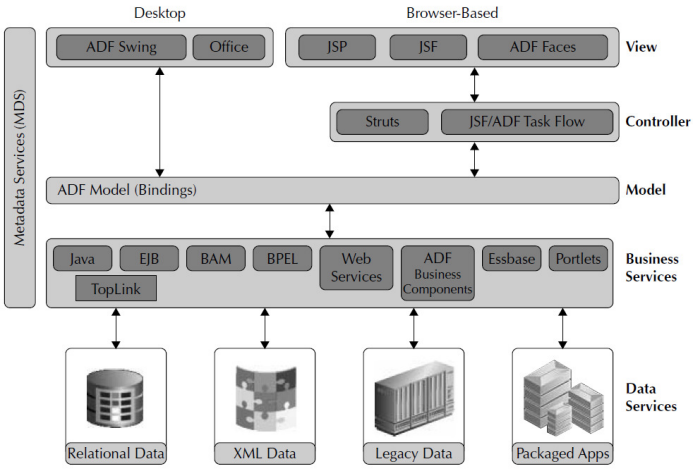
Kao alat za razvoj softverskog rešenja autori su koristili Jdeveloper, koji predstavlja besplatan IDE i koji je razvijen od strane Oracle korporacije. Nudi mogućnost za razvoj softverskih rešenja u Javi, XML-u, SQL i PSQL, HTML, Javascript, BPEL, PHP.

Pokriva ceo razvojni ciklus od dizajna preko kodiranja, debugging, optimizacije, profilisanja i deploy-a. Jdeveloper se integriše sa Oracle-ovim aplikacionim razvojnim framework-om ADF, koji predstavlja uopšteno end-to-end Java EE zasnovani framework koji još više pojednostavljuje razvoj aplikacije. Ovaj alat je u svojoj biti napravljen tako da krajnji korisnici, razvojni inženjeri budu što je moguće više produktivnji. Jedna oblast od naročitog značaja jeste podrška za ADF.

Oracle ADF je meta-framework koji zadovoljava ključne zahteve koji se predstavljaju pred framework.

MVC je često korišćeni dizajn patern koji razdvaja aplikacioni kod u tri nivoa (vidi sliku 2). Model koji definiše i validira podatke koje koristi aplikacija, *View* koji predstavlja korisnički interfejs koji prikazuje podatke iz sloja *Model* a takođe je u stalnoj interakciji sa slojem *Controller* tako što od njega dobija naredbe da ažurira korisnički interfejs kao i da ga izveštava kada se desi neki korisnički događaj (*user event*), *Controller* kod u ovom sloju određuje šta je reakcija kada se desi neki korisnički događaj u *View* sloju.

ADF je koristan framework ali mogućnost da se koristi nekoliko različitih rešenja u okviru svakog funkcionalnog dela može dovesti do problema – postojanje previše izbora. Što se tiče integracije sa bazom ADF nudi dva primarna izbora za direktno mapiranje baze podataka i Java koda: ADF *business* komponente i *Enterprise Java Bean* koristeći Java *persistence* API. [8]



Slika 1. Model View Controller pattern

C. *Jasper Reports*

Jasper Reports je veoma fleksibilan alat za izveštavanje koji ima sposobnost da isporuči bogat sadržaj na ekran, štampač, PDF, HTML ili XML fajl. Ova biblioteka je celokupna pisana u Javi i može se koristiti u raznim aplikacijama koje je podržavaju uključujući i J2EE i web aplikacije.

Vrši organizaciju podataka u skladu sa izabranim dizajnom izveštaja definisanim u XML fajlu. Da bi se mogao napuniti izvešaj potrebno ga je prvo kompajlirati. Kroz ovaj postupak objekat izveštaj je kreiran a zatim i serializovan ne bi li bilo omogućeno čuvanje na disku ili slanje preko mreže. Da bi napunili izveštaj *engine* treba da prihvati podatke koji će se koristiti u izveštaju. Neki od ovih podataka mogu biti prosleđeni kao parametri izveštaja ali se većina može naći u report data source-u. Reporting *engine* može direktno da

prihvati specijalne objekte sa izvora podataka iz kojih izvlači informacije da stavi na izveštaj ili može da samostalno radi sa JDBC objektom konekcije ukoliko su podaci u relacionoj bazi podataka. Kao rezultat svega je izveštaj koji je spreman za štampanje, koji je takođe serijalizovan za skladištenje ili slanje preko mreže a takođe se može direktno pregledati koristeći ugrađen Jasper Report Viewer ili može biti eksportovan u neke druge popularne formate kao što su PDF, HTML ili XML.

Jasper reports koristi SAX 2.0 API da parsira XML fajlove. U većini slučajeva kada se koristi Jasper report biblioteka radi se samo na nekoliko klasa i nije potrebno poznavati celokupan API.

Kao podrška ovoj biblioteci autori su koristili alat koji se zove IReport koji omogućava korisnicima Jasper-a interfejs za kreiranje dizajna izveštaja. Predstavlja open source alat koji može da kreira kompleksne izveštaje iz bilo koje Java aplikacije kroz Jasper report biblioteku [10].

III. PROJEKTOVANJE BAZE PODATAKA

U softverskom inženjerstvu Entity Relationship Model (ERM) predstavlja apstraktnu i konceptualnu reprezentaciju podataka. ERM je metoda za modelovanje baze podataka koja za rezultat daje konceptualnu šemu koja predstavlja semantički bogat model podataka sistema. Dijagrami koji se kreiraju u okviru ove metode nazivaju se Entity Relationship Diagrams, ER dijagrami, koji predstavljaju pogodnu dijagramsku tehniku za predstavljanje statičke strukture realnog sistema. On prevashodno uživa popularnost zbog dijagramskog načina prikaza šeme baze podataka.

Osnovni koncepti ER dijagrama su: tip Entiteta, tip Poveznika, domen, obeležje i svaki od njih ima svoj način označavanja na dijagramu.

Postoje dva nivoa detaljnosti prikaza ER dijagrama:

- Nivo naziva tipova – globalni nivo prikaza i
- Nivo naziva obeležja – detaljni nivo prikaza

Dijagrami klasa se široko koriste za iskazivanje modela statičke strukture sistema, šeme baze podataka i podšema. Ovi dijagrami koriste elemente kao što su klase i paketi. Dijagrami klasa takođe prikazuju i veze: asocijacija, zavisnost, realizacija, generalizacija. Dijagrami klasa se najviše koriste prilikom dizajniranja sistema zbog toga što se na njemu nalaze predstavljene sve klase koje će biti korišćene u budućem informacionom sistemu. Na osnovu ovoga olakšan je postupak određivanja statičkog odnosa koji postoji između ovih objekata [9].

Postupak projektovanja konceptualne šeme baze podataka se može sprovesti na osnovu dva pristupa. Prvi pristup predstavlja postupak potpune integracije eksternih šema. Podšema ili eksterna šema je model dela statičke strukture realnog sistema dobijen na osnovu dela šeme baze podataka koji je potreban i dovoljan za realizaciju zadataka jednog ili grupe transakcionih programa, sličnih sa stanovišta modelovanih procesa poslovanja i korisničkih zahteva. Prilikom postupka projektovanja eksternih šema bitno je jasno definisati grupe korisnika budućeg informacionog sistema. Za svaku od identifikovanih grupa korisnika projektuje se jedna eksterna šema. Nakon toga sledi faza njihove integracije u konceptualnu šemu. Postupak postupne integracije predstavlja neminovnost ukoliko se projektuju baze podataka sa velikim brojem koncepata.

Drugi pristup je direktni postupak projektovanja konceptualne šeme. Korisnički zahtevi identifikovani u fazi analize se spajaju u jedan skup zahteva na osnovu kojeg se projektuje jedna šema baze podataka. Nakon toga vrši se projektovanje eksternih šema, za svaku od korisničkih grupa i planiranih primena baze podataka. Postupak projektovanja eksternih šema se vrši polazeći od konceptualne šeme, izdvajanjem onih koncepata koji su bitni za obavljanje posmatranog zadatka za koji se eksterna šema oblikuje [3][7][6].

Prilikom projektovanja konceptualne šeme baze podataka automatizovanog sistema za podršku sistemu za upravljanje proizvodnim nalozima primenjen je direktni postupak, pre svega zbog mogućnosti sagledavanja problematike planiranja proizvodnje i pripreme procesa rada u celini. Dobijena šema baze podataka uključuje i koncepte koji ne pripadaju direktno upravljanju proizvodnim nalozima ali su neophodni za realizaciju. Projektovanje konceptualne šeme baze podataka u Oracle razvojnom okruženju je urađeno korišćenjem alata Entity Relatioship Diagrammer. [3].

IV. RAZVOJ PROGRAMSKE PODRŠKE

Razvoj programske podrške informacionog sistema započinje modeliranjem programske specifikacije modula. Programski moduli se definišu na osnovu izabranih skupova šema relacija koristeći se prethodno oblikovanom implementacionom šemom baze podataka. Šeme relacija jednog skupa, na osnovu kog se moduli oblikuju, grupišu se u komponente programskog modula, pri čemu jedna komponenta modula treba da predstavlja jednu logičku celinu za prezentaciju podataka. Broj komponenti u specifikaciji programskog modula može biti različit i zavisi od funkcije samog modula [3].

Jdeveloper podržava mogućnost automatskog generisnja View objekata na osnovu implementacione šeme baze podataka. Ovi objekti predstavljaju poslovne komponente koje prikupljaju podatke iz samog izvora podataka (npr. Baze podataka) i oblikuju ih da budu pogodni za upotrebu od strane klijenata, kao i da klijenti imaju mogućnost da ih menjaju. Na primer view objekat može da prikupi sve informacije neophodne za popunjavanje jednog elementa tabele u formi, vrši kreiranje i obradu unosa ili editovanje forme kao i kreiranje List of Values za popunjavanje padajućih lista.

Na osnovu kreiranih View objekata vrši se generisanje aplikacionih modula. Oracle ADF application modules su poslovne komponente koje reprezentuju određeni aplikacioni zadatak. Aplikacioni modul obezbeđuje model podataka za traženi zadatak tako što akumulira instance View objekta i View linka potrebnih za njegovo izvršavanje. Takođe poseduje usluge koje pomažu korisniku da izvrši posao, kao što su na primer: ažuriranje određenih informacija i slično. Moduli sadrže instance View objekta koji preuzimaju podatke koji su

bitni za korisnika a instance View linka definišu odnos između instance objekata. Ove instance mogu zajedno biti predstavljene putem stabla, koje se zove model podataka aplikacionog modula.

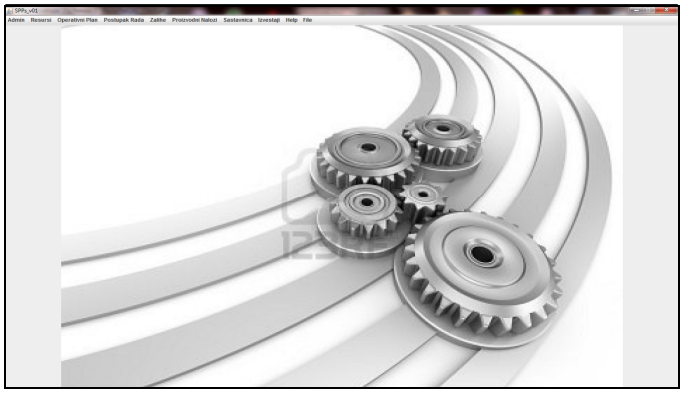
Takođe je moguće dodati instancu aplikacionog modula u drugu definiciju aplikacionog modula. Ovaj process koji se naziva ugnježdavanje aplikacionih modula, pruža mogućnost ponovnog korišćenja manjih aplikativnih celina. Na taj način moguće je značajno povećati efikasnost izrade aplikacije, kao i njihov kvalitet. Sledeći korak je izrada klijentske strane aplikacije u okviru kojeg se između ostalog specifcira konačni izgled korisničkog interfejsa [8].

V. PROTOTIP APLIKACIJE ZA PODRŠKU UPRAVLJANJU PROIZVODNIM NALOZIMA

Razvijeni prototip aplikacije za planiranje i pripremu procesa rada se sastoji od više ekranskih formi grupisanih u kategorije: Admin, Resursi, Operativni plan, Postupak rada, Zalihe, Proizvodni nalozi, Sastavnica. Sve ove forme su zajedno grupisane na glavnoj stranici pa samim tim navigacija kao i izbor određene forme vrši se izborom određene stavke menija.

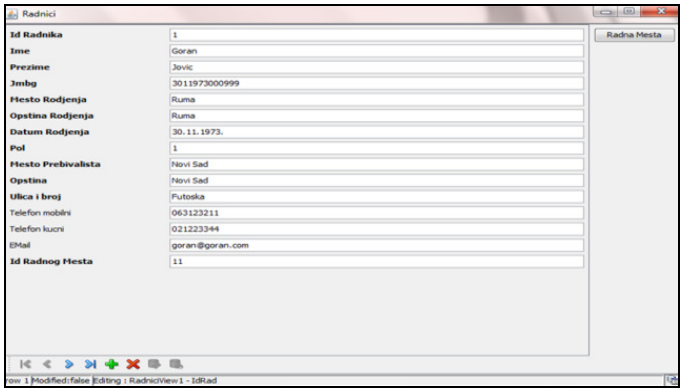
Na primer administracija radnika obavlja se kroz ekransku formu Radnici, koja obuhvata mehanizme za vođenje evidencije radnika u sistemu, gde je moguće izvršiti ubacivanje novog radnika, brisanje i modifikaciju postojećih radnika. Unos novog radnika se obavlja klikom na ikonicu koja ima izgled **+** a brisanje klikom na ikonicu **x.** Takođe prilikom unosa novog radnika imamo jednu funkcionalnost na ovoj formu koja to umnogome olakšava, na primer kada želimo da za novog radnika unesemo radno mesto dovoljno je da kliknemo na dugme **RadnaMesta** koje dalje otvara jedan iskačući prozorčić u okviru kojega se nalaze nabrojana sva radna mesta koja postoje u preduzeću i izborom jednog popunjava se ciljno polje na ekranskoj formi za evidenciju radnika, samim tim korisik ovog IS koji unosi novog radnika ne mora da zna napamet sva radna mesta koja postoje niti da troši vreme da ih traži po drugim ekranskim formama.

Pored postojanja ekranskih formi aplikacija poseduje i mogućnost izrade izveštaja. Sami izvestaji se generisu kako je već ranije navedeno pomoću Jasper Reports i Java koda u okviru Jdevelopera. Dovoljno je u okviru menija aplikacije izabrati stavku **izvestaji** i unutar nje izabrati željeni izvestaj, gde se na primer u slucaju resursa traži da se unese Id željenog resursa i aplikacija generiše PDF izveštaj na osnovu *real time* stanja u bazi podataka. Izveštaj se kreira na sledeći način, prvo imamo XML izvor koji treba da se kompajlira u .jasper fajl. Nakon toga potrebno je da obezbedimo podatke koji će da se ispišu na izveštaju. Da bi ovo odradili moramo da predstavimo podatke pomoću specifičnog jasper interfejsa koji se zove JRDataSource, ***Data Source + Jasper fajl = Print.*** Ovakav rezultat dalje je moguće eksportovati u željeni format.



Slika 2. Glavna strana aplikacije

Slika 3. Izveštaj aplikacije



Slika 4. Ekranska forma aplikacije

VI. ZAKLJUČAK

U postupku razvoja softverskog rešenja za podršku upravljanju proizvodnim nalozima autori su na najbolji mogući način težili da odgovore na sve postavljene zahteve i potrebe do kojih se došlo nakon detaljne analize potreba sistema. Samim tim sto analiza nije bila usmerena na jedan konkretan sistem već na više savremenih industrijskih sistema ovakva jedna aplikacija može da zadovolji osnovne potrebe bilo kog savremenog industrijskog sistema.

Nakon razvijenog prototipskog rešenja da bi došli do konačnog rešenja vršila se konstantna evaluacija informacionih potreba i stalno prilagođavanje funkcionalnosti aplikacije novonastalim potrebama. Nakon svega kao rezultat je softversko rešenje koje može da obezbedi povećanje produktivnosti, smanjenje vremena trajanja i troškove pripreme procesa rada, a samim tim i povećanje ukupne efikasnosti planiranja proizvodnje i proizvodnog sistema u celini.

ZAHVALNICA

Objavljivanje ovog članka realizovano je u okviru projekta „Unapređenje konkurentnosti Srbije u procesu pristupanja Evropskoj Uniji", br. 47028, podržano od strane Ministarstva prosvete, nauke i tehnološkog razvoja Republike Srbije, za period 2011.-2014. godine.

LITERATURA

[1] Shari Lawrence Pfleeger, Joanne M. Atlee, "Softversko inženjerstvo, teorija i praksa", Računarski fakultet Beograd, Computer Equipment and Trade, 2006.

[2] B. B. Agarwal, M. Gupta, S. P. Tayal, "Software Engineering and Testing", Jones and Bartlett Learning, 2009.

[3] D. Stefanović, C. Krsmanović, M. Rakić-Skoković, Đ. Tasković,"Jedan pristup planiranju efektivne proizvodnje i pripreme procesa rada", INFOTEH-JAHORINA Vol. 11, 2012.

[4] M. Weisfeld, "Objektno otijentisani način razmišljanja", CET, Beograd, 2003.

[5] B. Eckel, "Misliti na Javi", Mikro knjiga, 2002.

[6] B. Lazarević, Z. Marjanović, N. Aničić, S. Babarogić, "Baze Podataka", Univerzitet u Beogradu, Fakultet organizacionih nauka, 2003.

[7] P. Mogin, I. Luković, "Principi baze podataka", Fakultet tehničkih nauka, Novi Sad, 1996.

[8] D. Mills, P. Koletzke, A. Roy-Faderman, "Oracle Jdeveloper 11g handbook", Oracle Press, 2010.

[9] "OMG Unified Modeling Language", Superstructure, 2010, pristupljeno 2013.
http://www.omg.org/spec/UML/2.3/Superstructure/PDF/

[10] Jasper Reports Documentation, pristupljeno 2013.
https://www.jaspersoft.com/store/jasperreports-ultimate-guide-documentation

ABSTRACT

The real value of the applications that are used to support information systems is reflected in two main things. First you need to ensure smooth functioning in order to end-user enables to easily and effectively perform their tasks, and the second and most important is to provide access to important reports and accounts that will help the management to make better business decisions. The success of production and business systems in modern business conditions, among others, is dictated by type and speed of the system response to changing markets. Business processes must be implemented in the shortest possible time and resource consumption must be kept to a minimum. This work describes the design and developement of a working prototype to support management of production orders in Oracle development environment that includes modeling of functional structure, conceptual and implementation data base schema and the developement of screen forms and reports.

THE DEVELOPMENT OF SOFTWARE SOLUTION TO SUPPORT MANAGEMENT OF PRODUCTION ORDERS IN INDUSTRIAL PRODUCTION

Bojan Stojković, Darko Stefanović, Dejan Rašić, Goran Pilipović, Slavica Mitrović