UDC: 004.4 Info M: str. 11-16

PREGLED I KOMPARACIJA PRISTUPA ZA DETEKCIJU I ANALIZU GREŠAKA U SPREDŠITOVIMA
AN OVERVIEW AND COMPARISON OF SPREADSHEET ERROR DETECTION AND DEBUGGING APPROACHES

Lena Đorđević, Danica Lečić-Cvetković

REZIME: Spredšitovi se primenjuju u širokom opsegu organizacionih funkcija za realizaciju različitih aktivnosti. Koriste se za finansijske proračune, planiranje, agregaciju podataka, donošenje odluka i u druge svrhe. Ipak, brojna istraživanja ukazuju da su ovi programi izrazito podložni nastanku grešaka. Kao neki od najznačajnijih uzroka nastanka grešaka u spredšitovima mogu se izdvojiti razvoj modela od strane krajnjih korisnika i nepostojanje procesa obezbeđenja kvaliteta (OK) softvera. U skladu sa tim, predloženi su različiti pristupi usmereni na podršku krajnjim korisnicima u razvoju i korišćenju spredšit modela višeg kvaliteta. U ovom radu se predstavlja pregled literature na temu pristupa za detekciju i analizu grešaka u spredšitovima, od vizualizacije spredšitova, statičkih analiza i izveštaja, automatske detekcije i ispravke grešaka, testiranja, modelom vođenih pristupa razvoja, do pristupa projektovanja i održavanja. Na osnovu pregleda izabranog skupa pristupa, predlažu se mogući pravci budućih istraživanja.

KLJUČNE REČI: Spredšitovi, greške, detekcija grešaka, analiza grešaka.

ABSTRACT: Spreadsheets are applied in wide range of organizational functions for realization of different activities. They are used for financial calculations, planning, data aggregation, decision making, etc. However, a number of researches have shown that such programs are particularly error-prone. Some of the most important causes for spreadsheet error occurrence are that spreadsheet models are developed by end users and that standard software quality assurance (QA) processes are usually omitted. Correspondingly, researchers proposed a number of approaches aimed at supporting end users in developing and using the error-free spreadsheet models. In this paper, we provide a literature review on the topic of error detection and debugging approaches, including spreadsheet visualization, static analysis and quality reports, automated fault localization and repair, testing, model-driven development approaches, design and maintenance support. Based on this review, we outline directions for future research.

KEY WORDS: Spreadsheets, errors, error detection, debugging.

1. UVOD

Spredšit aplikacije, danas najčešće bazirane na *MS Excel* softverskom alatu, imaju široko polje primene. Ove interaktivne računarske aplikacije postale su popularne 80-ih godina dvadesetog veka, kao najpoznatija paradigma programiranja od strane krajnjih korisnika. Njihova glavna prednost je mogućnost, koju pružaju ekspertima iz različitih oblasti poslovanja, da kreiraju sopstvene alate za specifične probleme, u koje će ugraditi svoje ekspertsko znanje. Vreme potrebno za kreiranje takvih aplikacija je značajno kraće nego vreme potrebno za izradu klasičnih poslovnih aplikacija, koje se prave u okviru *IT* odeljenja preduzeća, u skladu sa standardima kompanije o obezbeđenju kvaliteta. Opisane karakteristike spredšitova omogućavaju krajnjim korisnicima brzu i jednostavnu izradu modela i aplikacija, ali ovo može biti i uzrok čestog nastanaka grešaka, lošeg dizajna modela i aplikacija. Greške se, usled nedovoljne obučenosti korisnika, lako prave, ali teško uočavaju. Eksperimenti prikazani u (Powell, Baker & Lawson, 2008) pokazali su da su spredšitovi podložniji nastanku greška nego ostali softveri. U okviru Evropske interesne grupe za rizik u spredšitovima (*EuSpRiG*[1]) prikupljeni su i predočeni značajni dokazi o slučajevima grešaka u spredšitovima koje su prouzrokovale velike finansijske gubitke u kompanijama. Ovi slučajevi pokazuju uticaj koji greške u spredšitovima mogu imati na poslovanje kompanija. Međutim, za razliku od procesa ili softvera koji su u organizacijama validirani i kontrolisani, za spredšitove ne postoje odgovarajuće procedure, ni standardi. Rizik pojave grešaka u spredšitovima izaziva potrebu za unapređenjem metoda i alata za njihovu detekciju. Svest o riziku ovog tipa se značajno povećala u poslednje dve decenije. U skladu da tim, mnogi autori razvijaju metodologije, tehnike i automatizovane alate za otkrivanje grešaka, koje bi krajnji korisnici mogli da primenjuju.

Nakon uvoda, u nastavku ovog rada biće prikazane različite taksonomije grešaka u spredšitovima. U trećem poglavlju predstavljene su kategorije pristupa za detekciju i analizu grešaka u spredšitovima, dok je u četvrtom poglavlju dat uporedni prikaz pristupa za detekciju i analizu grešaka u spredšitovima, u okviru navedenih kategorija. Peto poglavlje rada sadrži zaključak i pravce budućih istraživanja koji su definisani na osnovu prethodno prikazanog pregleda i analize.

2. TAKSONOMIJE SPREDŠIT GREŠAKA

Klasifikaciona šema spredšit grešaka treba da obuhvata najčešće i najvažnije greške. Efikasnost prevencije grešaka i tehnike detekcije mogu se proceniti na osnovu taksonomije grešaka koja obuhvata tipove, učestalost i moguće uzroke (Ayalew, 2001). Ipak, u skladu sa brojnim klasifikacijama definisanim od različitih autora, može se zaključiti da ne postoji univerzalni način za kategorizaciju grešaka. Greška može pripadati različitim kategorijama, u zavisnosti od stava programera i uzroka greške. U skladu sa važnošću spredšit grešaka, ova tema je često razmatrana, što se može videti u brojnim radovima: (Rajalingham, Chadwick, & Knight, 2000), (Howe

[1] http://www.eusprig.org/

& Simkin, 2006), (Caulkins, Morrison, & Weidemann, 2006), (Powell, Baker, & Lawson, 2008), (Powell, Baker, & Lawson, 2009), (Panko & Aurigemma, 2010) i drugi.

Jedna od prvih klasifikacija različitih tipova grešaka u spredšitovima, bazirana na dotadašnjem pregledu literature, prikazana je u (Ronen, Palley, & Halverson, 1989). U radu se prikazuju kategorije koje se odnose na sam model, kao što su greške u logici i one koje se odnose na izvršenje modela, kao što je pogrešna referenca ćelije.

Rad (Panko & Halverson, 1996) usmeren je na kompletan model spredšit grešaka i razdvajanje kvalitativnih i kvantitativnih grešaka. Kvantitativne greške su numeričke greške, koje odmah uzrokuju pogrešan rezultat. Kvalitativne greške ne vode odmah pojavljivanju greške, ali je mogu prouzrokovati, kada se spredšit menja. Prema (Panko & Halverson, 1996) tri glavne kategorije grešaka su:

- Mehaničke greške - nastaju kao greške kucanja, selektovanja ili druge jednostavne omaške;
- Logičke greške - nastaju kao rezultat izbora pogrešnog algoritma ili kreiranja pogrešne formule;
- Greške propusta - rezultat su izostavljanja dela modela i često vode pogrešnoj interpretaciji rezultata modela.

Isti izvor predstavlja istraživanje o greškama nastalim u razvoju i inspekciji spredšitova. Autori su kreirali taksonomiju istraživanja spredšit rizika u trodimenzionalnoj formi kocke. Tri strane kocke odnose se na problem istraživanja, fazu životnog ciklusa i metodologiju (eksperiment, anketa itd.) za istraživanje. Problem istraživanja odnosi se na strukturu, stvarne greške, iskustvo korisnika, pretpostavke i karakteristike spredšit modela (veličinu, procenat ćelija sa formulama ili podacima, kompleksnost formula, frekvenciju upotrebe spredšita (jednom ili više puta), broj ljudi koji ga koristi, namenu) i kontrolne politike.

Taksonomija ovih autora je revidirana i proširena u (Panko & Aurigemma, 2010). Svrha rane taksonomije je bila da podrži kvantitativna istraživanja i pokaže da su kvantitativne greške u spredšitovima česte, teške za detektovanje, kao i da su spredšitovi sa greškama čest slučaj u praksi. Revidirana taksonomija uvodi sledeće izmene:

- Postoji razlika između slučajnih grešaka i prekršaja korporativne prakse;
- Razlika između logičkih, mehaničkih i greška propusta zamenjena je češće korišćenom razlikom između domenskih i grešaka planiranja izraza u spredšitu sa jedne strane, i greška implementacije (omaški i propusta) sa druge. Greške planiranja predstavljaju pogrešnu implementaciju planova;
- Domenske greške planiranja pojavljuju se kada kreatori naprave greške u domenskom poznavanju modela (finansije, ekologija, fizika itd.). Greške planiranja izraza u spredšitu dešavaju se kada kreator planira pogrešan izraz za domenski algoritam;
- Logičke greške postaju greške zamene, a mehaničke se dele na omaške i propusta. Omaške su senzomotorne greške, kao što su greške u kucanju ili selektovanju. Propusti su greške koje nastaju kao posledica nenamernog izostavljanja elemata spredšit modela od strane kreatora.

U radu (Rajalingham, Chadwick, & Knight, 2000) predstavlja se taksonomija spredšit grešaka koja se zasniva na razlici između kvalitativnih i kvantitativnih grešaka. Za razliku od taksonomije prikazane u (Panko & Halverson, 1996), ovde se u daljoj distinkciji pojavljuju slučajne greške i greške rasuđivanja. Značajan element u ovoj taksonomiji grešaka je razlika između kreatora spredšit modela i krajnjih korisnika. Autori u razmatranje uzimaju i greške koje nastaju u interpretaciji rezultata spredšita. Ukoliko model daje tačan rezultat, ali se on pogrešno interpretira (na primer usled loše postavljenih zaglavlja i formatiranja), javlja se podjednako važna greška koliko i greška kreatora. Za razliku od Panko i Halverson taksonomije, koja se zasniva na uzroku greške, ova taksonomija je opštija i opisna kao hijerarhijski sistem kategorija spredšit greška, baziran na zajedničkim karakteristikama i vezama. Ova taksonomija je revidirana u radu (Rajalingham, 2005). Revidirana taksonomija je žbunasta (engl. *Bushy*) i grana se na tri ili više alternative. Nedostatak ove klasifikacije je poteškoća da se odredi gde treba svrstati grešku, a može se desiti i da se greška svrstava u dve ili više klasa. Autor Rajalingham u istom radu predstavlja i "binarnu" taksonomiju, pri čemu binarni izbor u svakom koraku olakšava korišćenje klasifikacije.

3. KATEGORIJE PRISTUPA ZA DETEKCIJU I ANALIZU GREŠAKA U SPREDŠITOVIMA

Pristupi za detekciju i analizu grešaka u spredšitovima, vrlo često, se zasnivaju na primeni metoda i alata iz oblasti softverskog inženjerstva. Ipak, zahtevi koji se postavljaju za pristupe obezbeđenja kvaliteta (OK) spredšitova se razlikuju od onih namenjenih imperativnim jezicima. Interakcija između korisnika i spredšit okruženja se značajno razlikuje od načina na koji su programi imperativnih jezika kreirani. Na primer, korisnici, uglavnom, započinju razvoj spredšit modela nestruktuiranim inkrementalnim procesom. Na osnovu ulaznih podatka, korisnici odmah dobijaju povratnu vizualnu informaciju. Dakle, alat koji se razvija za korisnike spredšitova mora biti kreiran tako da podrži inkrementalni proces razvoja i obezbedi trenutne povratne informacije. Osim već spomenute razlike, proračun u spredšitovima je značajno drugačiji nego kod imperativnih jezika. Proračuni se zasnivaju na zavisnostima podataka u ćelijama, dok su pravila sadržana u formulama. Ova činjenica se mora uzeti u obzir pri definisanju mehanizama i alata za OK. Još jedna značajna razlika je što se spredšit programi ne zasnivaju samo na jednostavnim modelima proračuna, tj. na fizičkom rasporedu, već je prostorni raspored labela i formula značajno određen semantikom proračuna. Ove prostorne informacije mogu se koristiti za detekciju inkonzistentnosti susednih ćelija i određivanje verovatnoće da je formula semantički tačna, za automatsko zaključivanje na osnovu labela ili rangiranje preporuka za ispravku, kod pristupa detekcije grešaka zasnovanih na cilju (Erwig, 2009).

U obzir se mora uzeti i da kreatori spredšitova, u velikom broju slučajeva, nisu profesionalni programeri. Kreatori imperativnih programa pohađaju formalne treninge, edukovani su u oblasti razvoja softvera i svesni su važnosti sistemskih procesa OK. Kreatori spredšitova, često, imaju ograničeno interesovanje i svesnost o značaju aktivnosti OK, kao što je testiranje.

Zbog toga, metodologije i alati moraju biti jednostavni za razumevanje i primenu.

Prema (Jannach et al., 2014), pristupi OK spredšitova mogu se klasifikovati u dve osnovne kategorije, u zavisnosti od njihove uloge i upotrebe u životnom ciklusu razvoja:

1. *Pronalaženje i ispravljanje grešaka* - odnosi se na tehnike i alate koji su dizajnirani da pomognu korisnicima da detektuju greške, utvrde uzrok nastanka i predlože ispravku. Ove alate najčešće koriste kreatori, revizori ili recenzenti, u toku ili nakon izrade spredšit modela;
2. *Izbegavanje grešaka* - obuhvata tehnike i alate koji podržavaju kreiranje spredšitova koji nemaju greške. Oni se primenjuju u toku procesa razvoja spredšit modela.

Detaljnija kategorizacija postojećih pristupa automatizovanog obezbeđenja kvaliteta spredšitova, prema (Jannach et al., 2014), prikazana je u Tabeli 1.

Tabela 1. Osnovne kategorije pristupa automatizovanog OK spredšitova (Jannach et al., 2014)

N°	Pristupi automatizovanog OK spredšitova	Pronalaženje grešaka	Izbegavanje grešaka
1.	Pristupi zasnovani na vizualizaciji	x	x
2.	Statičke analize koda i izveštaji	x	x
3.	Pristupi testiranja	x	
4.	Automatska lokalizacija i ispravka grešaka	x	
5.	Pristupi razvoja zasnovani na modelima		x
6.	Projektovanje i podrška održavanju		x

Pristupi kategorija 1 i 2 mogu da se koristite za otkrivanje i izbegavanje greška. Na primer, dobra vizualizacija zavisnosti ćelija podstiče uočavanje problema. U isto vreme, vizualizacija se može koristiti za isticanje ćelija i oblasti koje imaju veliku verovatnoću da će se greška u njima javiti u budućnosti, na primer, kada postoje ponavljajuće strukture u spredšitu. Statičke analize mogu identifikovati već postojeće probleme, kao što je referenciranje praznih ćelija, ali i služiti kao indikatori potencijalnih problema, na primer, izlistavanjem formula koje su kompleksne. Pristupi iz kategorija 3 i 4, uglavnom, se odnose na problem nalaženja i ispravljanja grešaka, s obzirom da ih korisnici primenjuju da identifikuju postojanje problema ili da lokalizuju uzrok greške. Pristupi kategorija 5 i 6 namenjeni su izbegavanju grešaka, na primer, procesom refaktorisanje ili dodavanjem nivoa apstrakcije.

Pristupi zasnovani na vizualizaciji obezbeđuju korisniku poboljšano predstavljanje određenih aspekata spredšitova, na osnovu kojih se lakše sagledavaju i razumeju odnosi i zavisnosti ćelija ili blokova spredšita. Vizualizacija omogućava korisniku bržu detekciju anomalija i neregularnosti u spredšitu. *Statičke analize i izveštaji* bazirani su na statičkoj analizi koda, sa ciljem da ukažu kreatoru na potencijalno problematične oblasti spredšita. Primeri ovih tehnika uključuju "miris koda" ili detekciju klonova podataka, ali i grupe tipičnih tehnika korišćenih u komercijalnim alatima, koji detektuju povratne veze ili prave izveštaje o nereferenciranim ćelijama. *Pristupi zasnovani na testiranju* imaju za cilj da stimulišu i podrže kreatore da sistematično testiraju spredšit aplikacije u toku i nakon razvoja. Ovi alati uključuju mehanizme za testiranje slučajeva, automatsko generisanje slučajeva za testiranje ili analize pokrivenosti različitih slučajeva. *Pristupi automatske lokalizacije i ispravke grešaka* zasnivaju se na računarskim analizama mogućih uzroka grešaka ili neočekivanog ponašanja (algoritamsko otklanjanje grešaka). Oni obuhvataju uključivanje dodatnih ulaza od strane kreatora, u obliku slučajeva za testiranje ili provera tačnosti ćelija. Neki od ovih alata omogućavaju ispravku na osnovu "sugestija". *Pristupi razvoja zasnovani na modelima*, usvajaju ideju primene objektno-orijentisanih konceptualnih modela, kao i tehnike razvoja vođene modelom, koje su danas opšte poznate i primenjene u softverskoj industriji. Prednosti ovih pristupa su uvođenje dodatnih nivoa apstrakcije ili upotreba mehanizama za generisanje koda. *Projektovanje i podrška održavanju* predstavlja pristup koji kreatoru olakšava izradu strukture spredšit modela ili aplikacije bez grešaka. Pristup uključuje automatizovane alate za refaktorisanje, metode za izbegavanje pogrešne reference ćelija i rukovanje izuzecima. U Tabeli 2 prikazani su različiti pristupi u okviru svake kategorije automatizovanog obezbeđenja kvaliteta spredšitova (Jannach et al., 2014).

Tabela 2. Pristupi automatizovanog OK spredšitova, (Jannach et al., 2014)

N°	Kategorije pristupa automatizovanog OK spredšitova	Podkategorije pristupa automatizovanog OK spredšitova
1.	Pristupi zasnovani na vizualizaciji	1.1 Vizualizacija tokova podataka i zavisnosti 1.2 Vizualizacija povezanih oblasti 1.3 Vizualizacija zasnovana na semantici 1.4 Pristupi vizualizacije informacija
2.	Statičke analize koda i izveštaji	2.1 Zaključivanje na osnovu jedinica i tipova 2.2 Spredšitovi sumnjive strukture 2.3 Statičke analize i komercijalni alati
3.	Pristupi testiranja	3.1 Testiranje adekvatnosti i slučajeva 3.2 Automatsko generisanje test slučajeva 3.3 Testiranje na osnovu uslova 3.4 Razvoj spredšitova vođen testiranjem
4.	Automatska lokalizacija i ispravka grešaka	4.1 Rangiranje kandidata na osnovu tragova 4.2 Lokalizacija grešaka na osnovu ograničenja 4.3 Pristupi popravke
5.	Pristupi razvoja vođeni modelom	5.1 Deklarativni spredšit modeli 5.2 Spredšit šabloni 5.3 Objektno-orijentisani vizualni modeli 5.4 Relacioni spredšit modeli
6.	Projektovanje i održavanje	6.1 Upravljanje referenciranjem 6.2 Rukovanje izuzecima 6.3 Promene i evolucija spredšita 6.4 Refaktorisanje 6.5 Ponovna upotreba

4. PREGLED PRISTUPA ZA DETEKCIJU I ANALIZU GREŠAKA U SPREDŠITOVIMA

U kontekstu vizualizacije spredšitova predloženo je više pristupa, od strane različitih autora: (Brath & Peters, 2006), (Hodnigg & Mittermeir, 2008), (Kankuzi & Ayalew, 2008), (Clermont, 2008), (Ayalew, 2009), (Hermans et a., 2011), (Hermans,

2013), koji omogućavaju predstavljanje određenih aspekata spredšita u vizualnoj formi. Svrha vizualizacije je razumevanje spredšita sa aspekta ispitivanja anomalija i detekcije grešaka. Na osnovu pregleda relevantne literature, može se zaključiti da samo nekoliko radova prikazuje sistematičnu evaluaciju predloženih metoda. U većini slučajeva, validacija je ograničena na kvalitativne intervjue ili male skupove učesnika ispitivanja prototipa. Zbog toga je primenljivost pristupa za krajnje korisnike nedovoljno potvrđena. Moguća evaluacija tehnika vizualizacije uključuje konstrukciju spredšita, inspekciju ili vežbe, ali i pristupe posmatranja zasnovane na razmišljanju „na glas". Sa aspekta primenjenih alata, *MS Excel* uključuje veoma mali skup jednostavnih opcija za vizualizaciju, u cilju analize i detekcije grešaka u spredšitu. Mogu se vizualizovati zavisnosti ćelija strelicama, ili pravougaonicima u boji, koji se odnose na ćelije referencirane u formuli. Takođe, u slučaju da je narušeno neko od pravila pisanja formula ugrađenih u *MS Excel*, u uglu ćelije se pojavljuje zeleni trougao. Još jedna od opcija koja se nudi korisnicima *MS Excel*-a je podržana idejom „semantičkih" imena promenljivih, koja se dodeljuju ćelijama umesto adresa redova i kolona, čime je omogućeno povećanje razumljivosti spredšita.

Cilj pristupa automatizovanog OK spredšitova iz kategorije statičke analize koda i izveštaji (Abraham & Erwig, 2006), (Abraham & Erwig, 2007), (Chambers & Erwig, 2009), (Hermans et al., 2012), (Cunha et al., 2012) je identifikacija formula ili strukturnih karakteristika spredšita, koje se smatraju indikatorima potencijalnih problema. Tačnost ovih pristupa zavisi od kvaliteta heuristike ili metrike za detekciju grešaka koje se koriste za definisanje sumnjivih delova spredšita. Uopšteno govoreći, alati za statičke analize predstavljaju familiju metoda za detekciju grešaka, koje se mogu naći kod komercijalnih alata. Sistemi zasnovani na jedinicama ili tipovima usmereni su na zaključivanje o dodatnim tipovima potencijalnih problema i mogu se smatrati jednostavnijim semantičkim pristupima. Takve tehnike zaključivanja imaju potencijal da identifikuju različite klase grešaka, ali postoji mogućnost i da detektuju veliki broj „lažnih uzbuna". Evaluacije pristupa su izvedene korišćenjem *EUSES*[2] korpusa i spredšitova kreiranih od strane studenata i profesora. Potencijalno ograničenje primene *EUSES* korpusa je nepoznavanje semantike formula koje se ispituju. Zbog toga se ne može, sa sigurnošću, tvrditi da je formula zaista netačna, odnosno da je tehnika uspešno primenjena. Sa aspekta krajnjeg korisnika, rezultati statičkih analiza su jednostavni za razumevanje.

Jedan od najvećih problema programa krajnjih korisnika je što, obično, nisu rigorozno testirani. Iz tog razloga alati za podršku procesa razvoja spredšitova treba da omoguće kreiranje modela sa što manje grešaka (Burnett et al., 2003), (Kruck, 2006), (Carver et al., 2006), (McDaid et al., 2008). Ipak, u ovom kontekstu, komercijalni spredšit alati imaju ograničene funkcionalnosti. *MS Excel* obezbeđuje samo osnovne alate za validaciju - koji se odnose na definisanje tipova i vrednosti individualnih ćelija. Problem povezan sa alatima za testiranje odnosi se i na dizajn korisničkog interfejsa. Efikasnost generisanja test slučajeva i njihove adekvatnosti još uvek nije dovoljno ispitana. Još jedno od nedovoljno obrađenih pitanja odnosi se na ograničenu svesnost krajnjih korisnika o važnosti testiranja. Potrebno je analizirati na koji način spredšit kreatori zaista testiraju spredšitove i da li bi mogli da usvoje principe razvoja zasnovane na testiranju. Testiranje mutacijom (Abraham & Erwig, 2009) sastoji se od uvođenja malih promena u program i provere koliko takvih mutanata se može eliminisati određenim skupom testova. Mutacije se mogu koristiti za testiranje manuelno i automatski kreiranih test slučajeva. U širem kontekstu detekcije i eliminacije grešaka, mogu se koristiti za evaluaciju pristupa debagovanja.

Efikasnost tehnika debagovanja koje se zasnivaju na automatskoj lokalizaciji grešaka i popravci (Ruthruff et al., 2006), (Abraham & Erwig, 2008), (Abreu et al., 2012), (Hofer et al., 2013), (Jannach & Schmitz, 2014), (Đorđević, et al., 2015) ocenjene su korišćenjem protokola evaluacije kod kojih su greške veštački ugrađene u spredšit koji se razmatra. Evaluacija je pokazala da predložene tehnike vode ka uspešnom rangiranju kandidata za greške ili popravci grešaka, ili je moguće određivanje skupa potencijalnih uzroka grešaka. Ipak, spredšitovi korišćeni za eksperimente su, uglavnom, veoma mali, tako da je skalabilnost pristupa nedokazana. Pristupi zasnovani na ograničenjima često su ograničeni na male dimenzije ili celobrojne vrednosti. Za sve pristupe ovog tipa, kreatori spredšitova moraju da odrede željene izlazne vrednosti ili ćelije koje proizvode tačan/netačan izlaz. Samim tim, pristupi podrazumevaju da korisnik unapred zna koja vrednost rezultata treba da se dobije. To se javlja kao problem primene ovog pristupa, jer kreatori, često, nemaju takve informacije ili zadaju pogrešne vrednosti.

Modelom vođeni pristupi razvoja (Engels & Erwig, 2005), (Erwig et al., 2006), (Luckey et al., 2012), (Cunha et al., 2012) uvode dodatne sintaksne ili semantičke nivoe apstrakcije u proces razvoja spredšita. Ovi dodatni mehanizmi i konceptualizacija treba da pomognu smanjenju jaza između gotovog spredšit modela i realnog problema. Pristupi imaju za cilj i podizanje nivoa kvaliteta dizajna spredšit modela, smanjenje grešaka i lakše održavanje. Primena pristupa zasnovanih na razvoju modela donosi i izazove koji se mogu naći kod standardnih procesa razvoja softvera, kao što je problem koevolucije modela i programa. Dizajn i jezik za modeliranje, takođe, su značajni, jer se, često, mora praviti kompromis između ekspresivnosti i razumljivosti. Dodatno, kod spredšitova se javlja problem kreatora koji, uglavnom, nisu stručnjaci u oblasti informacionih tehnologija. Oni ne razumeju prednosti primene različitih alata i dugoročne prednosti apstrakcije i struktuiranosti spredšit modela. U obzir treba uzeti i da je jedan od glavnih razloga popularnosti spredšitova to što ne zahtevaju strukturirani, formalni proces razvoja, već *ad-hoc*, interaktivan proces razvoja prototipa. Pristupi ove kategorije, uglavnom nisu evaluiranni ili je izvršena preliminarna evaluacija. Stoga, oni moraju biti sistematičnije evaluirani, kao i ispitana njihova primenljivost i prihvatljivost za korisnike.

Pristupi projektovanja i održavanja (Bekenn & Hooper, 2008), (Harutyunyan et al., 2012), (Badame & Dig, 2012), (Harris & Gulwani, 2011) podrazumevaju primenu postojećih tehnika softverskog inženjerstva na spredšitove. Ipak, neophodno je zadržati karakteristike razvoja spredšita, kao i razumljivost za krajnje korisnike. Nasuprot tome, neki pristupi zahtevaju poznavanje netrivijalnih programerskih koncepata.

2 http://eusesconsortium.org/resources.php

Pošto krajnji korisnici obično nisu profesionalni programeri, primenljivost ovih pristupa u praksi je diskutabilna.

Prikazani pristupi, koji pripadaju različitim kategorijama automatizovanog OK spredšitova, evaluirani na različite načine, na osnovu metoda iz oblasti računarskih nauka i informacionih sistema, koje su prilagođene spredšitovima. U okviru spomenutih naučnih oblasti razvijeni su brojni standardi i protokoli za evaluaciju. U kontekstu evaluacije razmatrani su eksperimenti sprovedeni u laboratorijskim uslovima, empirijske studije bez korisnika ili teorijska analiza. U nekim slučajevima, efikasnost pristupa uopšte nije ispitana. U novijim radovima, može se zapaziti primena simulacije u laboratorijskim uslovima, kao najčešće korišćen način evaluacije pristupa. Za evaluaciju se, često, koriste slučajno izabrani spredšitovi iz *EUSES* korpusa, dok su spredšitovi iz prakse, koji se odnose na realne probleme, značajno ređe evaluirani. Kriterijum za izbor spredšita za analizu uglavnom nije definisan ili ne postoji. U cilju izvođenja empirijske evaluacije, greške su veštački ugrađene u spredšit, bez definisanih pravila o tipu i lokaciji greške. Eksperimenti uglavnom nisu detaljno ili nisu uopšte dokumentovani, što onemogućava kreiranja *banchmark*-a. Dodatno, skalabilnost pristupa nije ispitana za sve pristupe prikazanih kategorija automatizovanog OK spredšitova.

5. ZAKLJUČAK I PRAVCI BUDUĆIH ISTRAŽIVANJA

Postojeći pristupi za OK spredšit modela, prikazani u brojnim radovima iz ove oblasti, pokazali su se perspektivnim, ali nedovoljno ispitanim. Spredšitovi korišćeni za evaluaciju ovih pristupa obično su namenski napravljeni, nisu realni primeri i relativno su mali, tako da njihova primenljivost u realnim okolnostima, kao i skalabilnost, nije potvrđena. Stoga, može se zaljučiti da postoji značajan prostor za unapređenje pristupa za OK spredšitova. U najvećem broju slučajeva, detekcija i analiza grešaka zasniva se na osobinama spredšit aplikacija, ali i na idejama iz različitih oblasti, kao što su: Softversko inženjerstvo, Operaciona istraživanja i druge. Gotovo nijedan od postojećih pristupa ne uzima u obzir karakteristike problema i način modeliranja, koji značajno utiču na nastanak, ali i na mogućnost otkrivanja grešaka. U odnosu na pokazne primere ili slučajno izabrane spredšitove iz *EUSES* korpusa, najčešće korišćene za evaluaciju postojećih pristupa, budući pravci istraživanja moraju uključiti u razmatranje spredšitove korišćene za rešavanje realnih problema, sa jasnom specifikacijom elemenata modela. Kao što je u radu već spomenuto, potencijalna ograničenja primene *EUSES* korpusa podrazumevaju nepoznavanja semantike formula za koje se pretpostavlja da su pogrešne, kao rezultat primene određenog pristupa. Zbog toga se ne može sa sigurnošću tvrditi da je formula zaista netačna, odnosno da je pristup uspešno primenjen. Opštost postojećih pristupa za OK spredšit modela ima za cilj široku oblast primene. Međutim, kao negativna posledica ove karakteristike javlja se problem prilagođenosti specifičnim problemima, koji se najčešće modeliraju u spredšitovima. Česta primena spredšitova za modeliranje specifičnih problema, povezanih sa poslovanjem realnih preduzeća, može se objasniti nepostojanjem komercijalnih softverskih alata primenljivih u ovim situacijama, kao i cenom razvoja takvih softvera.

Kao primer realnog problema, modelovanog i rešavanog u spredšitu, pogodnog za primenu pristupa automatske detekcije i analize grešaka, autori ovog rada predlažu problem upravljanja zalihama. Upravljanje zalihama predstavlja jedan od najznačajnijih problema Operacionog menadžmenta. Neki od najčešće korišćenih metodoloških pristupa za rešavanje ovog problema su matematičko modeliranje i simulacija (Tomašević, Stojanović, & Simeunović, 2014). Jedna od mogućih metoda modelovanja problema upravljanja zalihama je diskretni dinamički simulacioni spredšit model. Diskretnim dinamičkim simulacionim spredšit modelom se na verodostojan način, mogu predstaviti i rešavati problemi upravljanja zalihama sa kompleksnim matematičkim aparatima, i oni se, na taj način, mogu relativno jednostavno primeniti u realnim sistemima (Kostić, 2009). Algoritam za detekciju i analizu spredšit grešaka bi u ovom slučaju trebao da uključi karakteristike problema, kao i pretpostavke i ograničenja definisana principom modelovanja dinamičkih diskretnih procesa upravljanja u spredšitu.

Nakon analize šest pristupa za obezbeđenje kvaliteta modela u spredšitovima, predstavljenih u radu, može se pretpostaviti da bi pristupi automatske detekcije i ispravke grešaka u spredšitovima bili najpogodniji za primenu na problemima upravljanja zalihama, predstavljenim kao dinamički diskretni modeli u spredšitovima. Koncept diskretnog objekta upravljanja omogućava definisanje zakona ponašanja, oblasti upravljanja, ciljnog funkcionala i metode rešavanja modeliranog problema upravljanja, u cilju dobijanja optimalnog rešenja. Opisivanje dinamike ovih sistema vrši se upotrebom diskretnih jednačina i nejednačina. Test slučajevi koji se koriste za lokalizaciju grešaka, na osnovu ograničenja, podržani su ovim konceptom, kroz izdvojenu oblast upravljanja, namenjenu simulaciji, sa domenom određenim ograničenjima modeliranog problema. Takođe, način modeliranja problema upotrebom diskretnih jednačina i nejednačina je u skladu sa konceptom prevođenja spredšita u problem zadovoljenja ograničenja (*CSP*) i omogućava dodatno sužavanje mogućih kandidata uzroka grešaka, čime bi se mogla unaprediti i računarska efikasnost pri implementaciji metode. Struktura spredšit modela dinamičkog diskretnog sistema u skladu je sa modelom konkretnog problema i usmerena na određivanje što bolje vrednosti ciljnog funkcionala, što bi moglo unaprediti dijagnosticiranje na osnovu modela (*MBD*), za određivanje ćelije koja teoretski može biti pravi uzrok uočenog i neočekivanog izlaza proračuna. Takođe, problemi upravljanja zalihama predstavljaju realne probleme, za koje se može odrediti opseg očekivanih izlaznih vrednosti, što se često javlja kao problem kod veštački kreiranih test slučajeva, gde se očekivani izlaz vrlo često ne može sa definisati. S obzirom da su opisani modeli zasnovani na matematičkim modelima, izlazna vrednost, kao uporedna vrednost (engl. *Benchmark*) za testiranje nekog novog pristupa detekcije i analize grešaka, bi se mogla odrediti u jednostavnijim primerima, kako manuelno, tako i drugim programom.

Kao budući pravac istraživanja u oblasti pristupa unapređenja kvaliteta spredšitova može se razmatrati i spajanje različitih tehnika za predupređenje, detekciju i analizu grešaka u hibridne sisteme. Razvoj ovakvog pristupa može uključiti i definisanje algoritma zasnovanog na heuristici, koji uzima u obzir karakteristike spredšitova ali i modela razvijenog u spredšitu, zahteve različitih tehnika i slično.

LITERATURA

[1] Abraham, R., & Erwig, M. (2006). AutoTest: A Tool for Automatic Test Case Generation in Spreadsheets. *IEEE Symposium on Visual Languages and Human-Centric Computing,* 43–50. Brighton, United Kingdom.

[2] Abraham, R., & Erwig, M. (2007). UCheck: A Spreadsheet Type Checker for End Users. *Journal of Visual Languages & Computing,* 18 (1), 71–95.

[3] Abraham, R., & Erwig, M. (2008). Test-driven goal-directed debugging in spreadsheets. *IEEE Symposium on Visual Languages and Human Centric Computing*, 131–138. Germany.

[4] Abraham, R., & Erwig, M. (2009). Mutation Operators for Spreadsheets. *IEEE Transactions on Software Engineering*, 35 (1), 94–108.

[5] Abreu, R., Riboira, A., & Wotawa, F. (2012). Constraint-based debugging of spreadsheets. *XV Ibero-American Conference on Software Engineering,* 1-14. Buenos Aires, Argentina.

[6] Ayalew, Y. (2001). Spreadsheet Testing Using Interval Analysis. PhD thesis. Klagenfurt University, Austria.

[7] Ayalew, Y. (2009). A Visualization-based Approach for Improving Spreadsheet Quality. *Warm Up Workshop for ACM/IEEE ICSE 2010,* 13–16. Cape Town, S. Africa.

[8] Badame, S., & Dig, D. (2012). Refactoring meets Spreadsheet Formulas. *28th IEEE International Conference on Software Maintenance*, 399–409. Trento, Italy.

[9] Bekenn, B., & Hooper, R. (2008). Reducing Spreadsheet Risk with Formula-DataSleuth. *Spreadsheet Risks Interest Group 9th Annual Conference (EuSpRiG 2009).* London, United Kingdom.

[10] Brath, R., & Peters, M. (2006). Excel Visualizer: One Click WYSIWYG Spreadsheet Visualization. *10th International Conference on Information Visualisation,* 68–73. London, United Kingdom.

[11] Burnett, M., Cook, C., Pendse, O., Rothermel, G., Summet, J., & Wallace, C. (2003). End-User Software Engineering with Assertions in the Spreadsheet Paradigm. *25th International Conference on Software Engineering*. Portland, OR USA.

[12] Carver, J., Fisher II, M., & Rothermel, G. (2006). An Empirical Evaluation of a Testing and Debugging Methodology for Excel. *5th ACM-IEEE International Symposium on Empirical Software Engineering*, 278-287. Rio de Janeiro, Brazil.

[13] Caulkins, J., Morrison, E., & Weidemann, T. (2006). Spreadsheet errors and decision making: evidence from field interviews. Journal of End User Computing, 19 (3), 1–23.

[14] Chambers, C., & Erwig, M. (2009). Automatic Detection of Dimension Errors in Spreadsheets. *Journal of Visual Languages & Computing,* 20 (4), 269–283.

[15] Clermont, M. (2004). A Toolkit for Scalable Spreadsheet Visualization. *Spreadsheet Risks Interest Group 5th Annual Conference* (*EuSpRiG 2004)*. Klagenfurt, Austria.

[16] Cunha, J., Fernandes, J., Ribeiro, H., & Saraiva, J. (2012). Towards a Catalog of Spreadsheet Smells. *12th International Conference on Computational Science and Its Applications,* 202–216. Salvador de Bahia, Brazil.

[17] Cunha, J., Fernandes, J. P., & Saraiva, J. (2012). From Relational ClassSheets to UML+OCL. *27th Annual ACM Symposium on Applied Computing,* 1151–1158. Trento, Italy.

[18] Đorđević, L., Ljubičić, M., Marjanović, Z., Antić, S. (2015). Primena tehnologije semantičkog web-a za detekciju grešaka u spredšit modelima diskretnih sistema, *XXI naučno-stručna i biznis konferencija YU INFO 2015*,121-126. Kopaonik, Srbija.

[19] Engels, G., & Erwig, M. (2005). ClassSheets: Automatic Generation of Spreadsheet Applications from Object-Oriented Specifications. *20th IEEE/ACM International Conference on Automated Software Engineering*, 124–133. Long Beach, USA.

[20] Erwig, M. (2009). Software engineering for spreadsheets. *IEEE Software*, 26 (5) , 25–30.

[21] Erwig, M., Abraham, R., Kollmansberger, S., & Cooperstein, I. (2006). Gencel:A Program Generator for Correct Spreadsheets. *Journal of Functional Programming* 16 (3), 293–325.

[22] Harris, W. R., & Gulwani, S. (2011). Spreadsheet Table Transformations from Examples. *The 32nd ACM SIGPLAN conference on Programming language design and implementation*, 317–328. San Jose, CA, USA.

[23] Harutyunyan, A., Borradaile, G., Chambers, C., & Scaffidi, C. (2012). Planted model evaluation of algorithms for identifying differences between spreadsheets. *IEEE Symposium on Visual Languages and Human-Centric Computing,* 7–14. Innsbruck, Austria.

[24] Hermans, F. F. (2013). *Analyzing and visualizing spreadsheets*. Netherlands: Software Engineering Research Group, Delft University of Technology.

[25] Hermans, F., Pinzger, M., & Deursen, A. (2011). Supporting Professional Spreadsheet Users by Generating Leveled Dataflow Diagrams. *33rd International Conference on Software Engineering*, 451–460. Waikiki, Honolulu, HI, USA.

[26] Hermans, F., Pinzger, M., & Deursen, A. (2012). Detecting Code Smells in Spreadsheet Formulas. *28th IEEE International Conference on Software Maintenance*, 409–418. Trento, Italy.

[27] Hodnigg, K., & Mittermeir, R. T. (2008). Metrics-Based Spreadsheet Visualization: Support for Focused Maintenance. *Spreadsheet Risks Interest Group 9th Annual Conference (EuSpRiG 2008),* 79–94. London, UK.

[28] Hofer, B., Riboira, A., Wotawa, F., Abreu, R., & Getzner, E. (2013). On the empirical evaluation of fault localization techniques for spreadsheets. *16th International Conference (FASE 2013), Held as Part of the European Joint Conferences on Theory and Practice of Software*, 68-82. Rome, Italy: Springer Berlin Heidelberg.

[29] Howe, H., & Simkin, M. G. (2006). Factors affecting the ability to detect spreadsheet errors. *Decision Sciences Journal of Innovative Education,* 4 (1), 101–122.

[30] Jannach, D., & Schmitz, T. (2014). Model-based diagnosis of spreadsheet programs: a constraint-based debugging approach. *Automated Software Engineering*, 1-40.

[31] Jannach, D., Schmitz, T., Hofer, B., & Wotawa, F. (2014). Avoiding, Finding and Fixing Spreadsheet Errors-A Survey of Automated Approaches for Spreadsheet QA. *Journal of Systems and Software*, Vol. 94, 129-150.

[32] Kankuzi, B., & Ayalew, Y. (2008). An End-User Oriented Graph-Based Visualization for Spreadsheets. *4th International Workshop on End-User Software Engineering*, 86–90. Leipzig, Germany.

[33] Kostić, K. (2009). Inventory control as a discrete system control for the fixed-order quantity system. *Applied Mathematical Modelling*, 33(11), 4201-4214.

[34] Kruck, S. E. (2006). Testing spreadsheet accuracy theory. *Information & Software Technology*, 48(3), 204-213.

[35] Luckey, M., Erwig, M., & Engels, G. (2012). Systematic Evolution of Model-Based Spreadsheet Applications. *Journal of Visual Languages & Computing,* 23 (5).

[36] McDaid, K., Rust, A., & Bishop, B. (2008). Test-Driven Development: Can it Work for Spreadsheets?. *4th International Workshop on End-User Software Engineering*, 25–29. Leipzig, Germany.

[37] Panko, R. R., & Halverson, R. P. (1996). Spreadsheets on trial: A survey of research on spreadsheet risks. *29th Annual Hawaii International Conference on System Sciences*, 326-336. Hawaii, SAD.

[38] Panko, R., & Aurigemma, S. (2010). Revising the Panko-Halverson taxonomy of spreadsheet errors. *Decision Support System*, 49(2), 235-244.

[39] Powell, S. G., Baker, K. R., & Lawson, B. (2008). A Critical Review of the Literature on Spreadsheet Errors. *Decision Support Systems*, 46(1), 128-138.

[40] Powell, S. G., Baker, K. R., & Lawson, B. (2009). Errors in Operational Spreadsheets. *Journal of Organizational and End User Computing*, 21(3), 24-36.

[41] Ronen, B., Palley, M., & Halverson, L. (1989). Spreadsheet analysis and design. *Communication of the ACM*, 32(1), 84-93.

[42] Rajalingham, K., Chadwick, D., & Knight, B. (2000). Classification of spreadsheet errors. *European Spreadsheet Risks Interest Group 1st Annual Conference (EuSpRIG 2000)*, 23–34. Greenwich, England.

[43] Rajalingham, K. (2005). A Revised Classification of Spreadsheet Errors. *European Spreadsheet Risks Interest Group 6th Annual Conference (EuSpRIG 2005),* 185-199. Greenwich, London.

[44] Ruthruff, J. R., Burnett, M., & Rothermel, G. (2006). Interactive Fault Localization Techniques in Spreadsheet Environment. *IEEE Transactions on Software Engineering*, 32 (4), 213–239.

[45] Tomašević, I., Stojanović, D., & Simeunović, B. (2014). Operations management research- An update for 21st century. *XIV international symposium SymOrg 2014: New business models and sustainable competitiveness*, 1280-1287. Zlatibor, Srbija.

M.Sc. Đorđević Lena, Fakultet organizacionih nauka, Univerzitet u Beogradu.
Kontakt: djordjevic.lena@fon.bg.ac.rs
Oblasti interesovanja: spredšit inženjerstvo, spredšit menadžment, upravljanje materijalnim i nematerijalnim tokovima, upravljački sistemi

Dr Danica Lečić-Cvetković, vanredni profesor, Fakultet organizacionih nauka, Univerzitet u Beogradu.
Kontakt: lecic.danica@fon.bg.ac.rs
Oblasti interesovanja: upravljanje proizvodnjom i pružanjem usluga, primena infomacionih i internet tehnologija u upravljanju proizvodnjom i uslugama