SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

DIPLOMSKI RAD

OPĆE INTELIGENTNO OKRUŽENJE ZA POUČAVANJE - GIFT TUTOR (Generalized Intelligent Framework for Tutoring)

Mensur Duraković

Mentorica: doc.dr.sc. Ani Grubišić

Split, siječanj 2016.

Sažetak

Ovaj rad donosi kratki pregled općeg inteligentnog okruženja za poučavanje tj. GIFT-a (eng. Generalized Intelligent Framework for Tutoring) i njegovu u smislu integracije C# aplikacije za poučavanje objektno-orijentiranog programiranja u programskom jeziku C#. U ovom radu će biti objašnjene osnovne funkcionalnosti GIFT okruženja poput izrade kurseva, alata za pomoć pri izradi istih, skripti za integraciju sa uređajima, drugim programskim jezicima i slično. Za kraj će biti objašnjene neke od osnovnih prednosti i mana za korištenje GIFT-a u nastavi. Opće inteligentno okruženje za poučavanje (eng. GIFT - Generalized Inteligent Framework for Tutoring) je „open-source“, modularno, korisnički orijentirano okruženje koje osigurava alate, metode i druge usluge dizajnirane u svrhu povećanja i unaprijeđenja integracije aplikacija u stvaranju inteligentnih i prilagodljivih tutorskih sustava. Ovaj rad daje kratki pregled o GIFT-u iz tehničke perspektive i opisuje ključne elemente potrebne kako bi se integrirala neka nova aplikacija u samo okruženje. Također opisana je i integracija C# aplikacije (čiji je cilj poučavanje objektno orijentiranog programiranja) u GIFT okruženje. Objektno orijentirano programiranje je u posljednjih nekoliko godina, postalo jedna od najutjecajnijih programskih paradigmi. Naširoko se koristi u obrazovanju i industriji, i gotovo svako sveučilište ima objektno orijentiranu paradigmu negdje u svom kurikulumu. Poučavanje objektno-orijentiranog programiranja i dalje je ostalo teško izvodivo u praksi. Tema ovog rada je upravo objasniti tu čudnu težinu objektno orijentiranog programiranja i korištenje prednosti GIFT-a za lakše učenje istog.

Summary

This diploma thesis provides a brief overview of GIFT (General Intelligent Framework for Tutoring) and its application in terms of the integration of C # application for teaching object-oriented programming. Here will be explained the basic functionality of GIFT enviroments such as making courses, tools to assist in the preparation of courses, scripts for integration with the devices and other programming languages. In the end, it will be explained some of the basic advantages and disadvantages of using GIFT in teaching. Generalized Framework for Intelligent Tutoring is "open-source", a modular, user-oriented environment that provides tools, methods and other services designed to increase and improve integration of applications in the creation of intelligent and adaptive tutoring systems. Here is provided a brief overview of the GIFT-in from a technical perspective and described the key elements required to integrate a new application in the environment itself. Also here is described integration of new C # application (which aims to teach object-oriented programming) in GIFT environment. Object-oriented programming in recent years, has become one of the most influential programming paradigms. Widely used in education and industry, and almost every university has an object-oriented paradigm somewhere in their curriculum. Teaching object-oriented programming still remains very difficult in practice feasible. The main goal of this diploma thesis is to explain the strange weight of object-oriented programming and take advantage of GIFT to learn it easier.

Sadržaj

Uvod

Poučavanje uz pomoć računala danas je u nastavi postalo normalna pojava. Postoje razni načini kako se računalo može iskoristiti u nastavi, a sve u svrhu kako bi olakšalo i pomoglo nastavniku u poučavanju kompleksnijih nastavnih cjelina ili jedinica. Raznorazni programi koji obavljaju različite zadatke i samim time nastavniku štede vrijeme kako u pripremi za nastavu, tako i tijekom nastave pri predavanju, objašnjavanju složenijeg gradiva, bolje vizualizacije i slično. Danas se najčešće koriste Microsoft Office alati, Adobe-ove aplikacije poput Adobe reader-a za čitanje PDF dokumenata, Adobe Photoshopa za crtanje, dizajn, grafičku obradu i slično. Pored ovih programa tu su i okruženja kao što su Moodle-a, Geenio, edX-platform, mySchoolApp i slično koji su danas jako zastupljeni i čiji je primarni cilj obrazovanje i pomoć u obrazovanju.

Okruženje slično prethodno navedenim okruženjima je i GIFT (eng. Generalized Intelligent Framework for Tutoring) koje će biti detaljno opisano u ovom radu. GIFT je jako „mlad“ alat američkog porijekla. S obzirom da se GIFT razlikuje od drugih okruženja po svojoj polivalentnosti i mogućnosti integracije drugih objekata unutar GIFT-a, posebna pažnja je posvećena integraciji C# aplikacije unutar GIFT okruženja.

Opće inteligentno okruženje za poučavanje (GIFT) je okvir i set alata za stvaranje inteligentnih i adaptivnih tutorskih sustava. U svojoj sadašnjoj formi GIFT je u velikoj mjeri R&D (eng. R&D - Research And Development) alat, dizajniran za pružanje fleksibilne eksperimentialne platforme za istraživače u inteligentnom i adaptivnom području poučavanja. Kako stvari stoje, GIFT je iz godine u godinu sve bliže da postane kvalitetan i prikladan okvir za uporabu u općim inteligentnim sustavima za učenja i poučavanje. Općenito govoreći, GIFT može predstavljati generički sadržaj kao što su dokumenti, multi-medijskih sadržaji, itd, međutim specijalizirani sadržaji su obično predstavljeni preko vanjskog sustava programske podrške, koji će se u ovom radu referirati kao aplikacije za poučavanje (eng. TA - Training application). GIFT pruža standardizirani način integriranja aplikacija za poučavanje i uključuje mnoge alate i usluge koje su potrebne za transformaciju aplikacija za poučavanje u jedan kompaktan inteligentni i/ili

adaptivni tutorski sustav. U tu svrhu zamišljeno je da se iskoristi ta prednost GIFT-a kako bi se kreirao jedan tutorski sustav za cijeli kolegij objektno-orijentiranog programiranja.

1. GIFT

GIFT je empirijski baziran, korisnički orijentiran skup alata, metoda i standarda napravljenih tako da se na što lakši način kreiraju, koriste i ažuriraju računalno-temeljeni sustavi za poučavanje (CBTS, eng. Computer Based Training System), upravlja procesom poučavanja i procijenjuje učinak prethodno navedenih računalno-temeljenih tutorskih objekata, komponenti i metodologije istih. GIFT je jedan od projekata koji se razvija u domeni projekata pod nazivom „Adaptive Tutoring Research Science & Technology“ u LITE laboratoriju (eng. Learning in Intelligent Tutoring Environments). Cijeli odjel spada pod ARL-HRED (U.S. Army Research Laboratory - Human Research and Engineering Directorate) sektor američke Vlade. GIFT se počeo upotrebljavati u praksi u 2011. godini od strane LITE Lab tima. Prva javna demonstracija GIFT-a je provedena na „Interservice/Industry Training Systems and Education“ konferenciji (I/ITSEC) u prosincu 2011. godine. Prva verzija GIFT-a završena je u svibnju 2012. godine. Dok se GIFT razvija kako bi se olakšalo korištenje CBTS-a od strane američke vojske, glavna namjera je da se zajednički razvija i ima funkciju kao temeljna jezgra za CBTS istraživanja koja se provode u okviru vlade, industrije i akademske zajednice. Do 03. studenog 2015. godine GIFT je bio dostupan samo kao desktop aplikacija s mogućnošću umrežavanja. Od 03. studenog 2015. godine dostupna je i web aplikacija GIFT-a kojoj se može pristupati na isti način kao i desktop aplikaciji. Istraživački ciljevi koje GIFT kao okruženje ima su:

- stvaranje ontologije za predstavljanje znanja i koncepata CBTS-a, njihovim odnosima i njihovim interakcijama;
- kreiranje automatiziranih alati i procesa dostupnih korisniku GIFT-a, vlastitih modela i komponenti;
- korištenje automatiziranih metoda isporuke prilagođene nastave za pojedince i male timove;
- implementacija skupa alata za višekratnu upotrebu, domene neovisnih modula koji uključuju standardne procese i strukture;
- uključivanje metodologije za potporu sustavne usporedbe i procjene CBTS sustava, njihovih dijelova i nastavnih metoda;

GIFT obuhvaća mnoštvo alata i usluga. Opće usluge i standardi GIFT-a uključuju:

- Standardni pristup za povezivanje sa aplikacija za poučavanje,
- Prikaz znanja,
- Ocjenjivanje izvedbe,
- Tijek predmeta,
- Pedagoški model (uključujući mikro i makro prilagodbe),
- Modeliranje orijentirano prema učeniku,
- Integrirana podrška za ankete (pitanja s autorskih alata),
- Sustav za upravljanje učenjem,
- Standardizirani pristup za integriranje fizioloških (i drugih) senzora

Još jedan ključni (ključan) aspekt GIFT-a je to da je "open source" sustav. Na razvoju GIFT-a trenutno radi američka kompanija Dignitas Technologies, ali ukoliko je potrebno, također i razvojna zajednica može dati svoj doprinos u unaprijeđenju GIFT-a. Rezultati iz sadašnjih i nadolazećih pokusa, kao što su pedagoški modeli, modeli orijentirani prema učeniku itd. mogu se ugraditi u budućim izdanjima GIFT-a, ukoliko neko od programera iz razvojne zajednice osmisli takvo što. Dakle, GIFT se stalno razvija i poboljšava svoj sustav, dok su pojedinačni doprinosi integrirani na obostranu korist svih korisnika u zajednici. GIFT okruženje sadržava servisno orijentiranu arhitekturu i sastoji se od više djelomično spojenih modula, koji komuniciraju putem asinkronih poruka koje prolaze preko zajedničkog dodatka. Kako GIFT obiluje modulima i funkcijama ovdje će biti opisani samo ključni moduli te njihove primarne funkcije (kojih ima mnogo). Neki važniji su:

- Modul za pristup (eng. Gateway module): Stvara vezu između GIFT-a i aplikacija za poučavanje (TA, eng. Training application),
- Modul senzora: Povezuje GIFT sa fiziološkim senzorima na standardizirani način,
- Modul orijentiran učeniku: Modelira kognitivni i afektivni element te performanse učenika,
- Pedagoški modul: Odgovoran za izradu domena neovisnih pedagoških odluka, primjenom internog modela na pedagoškom temelju stanja učenika,
- Domenski modul: Obavlja procjene uspješnosti, na temelju domenski generiranih pravila stručnjaka (autora domene), obavlja domenski specifične implementacije u

pedagoškim zahtjevima na temelju domene te (zajedno s pedagoškim modulom) nadzire tok kolegija,

- Tutor modul: Predstavlja sučelje za zadatke kao što su predstavljanje anketnih pitanja, pruža povratnu informaciju, uključen je u dijaloge i slično,
- Modul sustava za upravljanje učenjem (eng. LMS – Learning Managment System): predstavlja GIFT vezu s vanjskim sustavima za upravljanje učenjem, za čuvanje i održavanje učeničkih zapisa, biografskih podataka, materijala kolegija i slično,
- Modul sustavan za upravljanje korisnicima (eng. UMS – User Managment System): Uloga mu je upravljanje korisnicima GIFT sustava, upravlja sa anketama i anketnim podacima te osigurava funkcije za logiranje korisnika,
- Modul za nadgledanje (eng. Monitor module): Modul koji se koristi kao kontrolna ploča za pokretanje i zaustavljanje drugih GIFT modula te praćenje stanja aktivnih GIFT sesija.

Kako je glavna tema ovog rada integracija aplikacije za poučavanje izrađene u programskom jeziku C# unutar GIFT-a, potrebno je poznavati osnovne elemente arhitekture GIFT-a kako bi se uspješno obavila integracija navedene aplikacije. Potrebno je razumjeti koncepte poput GIFT poruka, vrste GIFT poruka, dodatak za gateaway modul, domain module, datoteke s područnim znanjem (eng. DKF - Domain knowledge Files), na koji način gore navedeni koncepti funkcioniraju i veze među istima.

1.1. GIFT poruke

Klase GIFT poruka: GIFT poruke su jedini način komuniciranja između GIFT modula. Hijerarhija jedne klase GIFT poruke se sastoji od tri najvažnije klase, od toga jedna je bazna klasa, preostale dvije su podklase. Bazna klasa GIFT poruke sadrži sva polja poput podatka za vrijeme obavljanja, "payload" tipa, predmetnu referencu za dodatni "payload" tip, polje za identifikaciju izvora i odredišnih modula, te mnoge druge. Preostale dvije podklase sadrže dodatna polja za GIFT kontekst, kao što su identifikator korisničke sesije i identifikator domenske sesije.

"Payload" tip poruka: Payload se može shvatiti kao neki podatak koji poruka nosi i koji se može javljati u različitom formatu. Za podršku među procesima komunikacije, poruke i njihove nosivosti moraju poštivati dogovoreno kodiranje i dekodiranje u sustavu. Od GIFT

verzije 3.0 kao zadana shema postavljena je JSON notacija (eng. JSON – JavaScript Object Notation). JSON se danas koristi u većini programskih jezika kod izrade API funkcija koje vraćaju određene podatke, lagan je za korištenje i ima svojstvo dobre komunikacije i sa drugim tehnologijama.

Vrste poruka: Svaka GIFT poruka ima pridruženi tip. Različite vrste poruka su navedene u .java klasi „`mil.arl.gift.common.enums.MessageTypeEnum`“ koja se može pronaći u direktoriju sa GIFT source datotekama.

1.2. Sučelje aplikacija za poučavanje

Postoje dva osnovna uvjeta koja aplikacija za poučavanje (eng. TA – Training application") mora ispuniti za zadovoljavajuću integraciju s GIFT-om. Prvi uvjet je sredstvo za prijenos stanja iz aplikacije za poučavanje na GIFT. Drugi uvjet je omogućiti GIFT-u odgovarajući stupanj kontrole nad aplikacijom za poučavanje. Osnovne kontrole, kao što su pokretanje aplikacija za poučavanje, učitavanje specifičnog sadržaja i zatvaranje aplikacije za poučavanje su vrlo korisne u izradi kvalitetnih rješenja, iako nisu uvijek sve potrebne u istom procesu.

Uvjet za komunikaciju stanja odmah je uspostavljen ako aplikacija za poučavanje u sebi ima implementiran objekt za komuniciranje putem standardiziranog mrežnog protokola kao što je DIS protokol (eng. DIS – Distributed Interactive Simulation). Za C# aplikaciju i komunikaciju sa GIFT-om u ovom radu je korištena XML-RPC.net biblioteka koja je zadužena za razmjenu poruka između aplikacije za poučavanje i GIFT okruženja, a sve u svrhu kako bi GIFT mogao sa sigurnošću znati kada treba pokrenuti aplikaciju unutar kolegija, koju reakciju treba imati nakon određenog događaja u aplikaciji za poučavanje i kada se treba zatvoriti aplikacija za poučavanje. Odabrana je baš XML-RPC.net bibilioteka za ovu ulogu jer je lagana za integraciju u aplikaciju, pruža jednostavnu sintaksu i jednostavno se uključi preko sučelja za ubacivanje biblioteka (u Visual Studiu). Još jedan od razloga je i taj što je u predefiniranom primjeru testne C# aplikacije koja dolazi sa instalacijom GIFT-a također korištena XML-RPC .net biblioteka.

Kontrola aplikacije za poučavanje od strane GIFT-a u usporedbi s prvim konceptom slijedi sličan obrazac. Ako postojeći protokol postoji, trebao bi biti korišten, ukoliko ne, onda će biti potreban prilagođeni razvoj (poput XML-RPC.net biblioteke). U odnosu na osnovne

poruke poput početnih, poruka učitavanja i poruka zaustavljanja neki slučajevi upotrebe integracija aplikacija za poučavanje mogu zahtijevati malo napredniju interakciju, međutim to nije toliko važno za ovu temu rada.

1.3. Dodatak za gateway modul

Proces prilagodbe aplikacija za poučavanje prema GIFT gateway modulu uključuje stvaranje dodatka (eng. plugin) za gateway modul. Za kreiranje ovog dodatka potrebno je osnovno programersko znanje jer je sve dobro opisano u developer dokumentaciji GIFT-a te je potrebno samo pratiti korake sa razumijevanjem i u kodu napraviti ono što se traži. Ukoliko se bavite programiranjem ovakvih dodataka, dobra je praksa uvijek provjeriti da li je jedan od postojećih dodataka pogodan za ponovnu uporabu kako se ne bi bez potrebe stvarao novi dodatak. Od GIFT verzije 3.0 uključeni su i neki integrirani dodaci za softvere poput: MS Power Point-a, TC3Sim-a i VBS2-a i drugih.

Čak i ukoliko je potreban novi dodatak, postojeći dodaci mogu poslužiti kao odlična referenca za izradu. Pri razvoju novog dodatka, primarni cilj je implementirati baznu .java klasu:

`mil.arl.gift.gateway.interop.AbstractInteropInterface`. Zahtjevi nove podklase su minimalni, ali pri implementaciji za svaku od apstraktnih metoda, dodatak će neprimjetno operirati unutar modula konteksta. Osim toga, u dodatak treba implementirati svaku potrebnu funkcionalnost zahtijeva, kao što je primanje poruka od aplikacija za poučavanje te njihovo pretvaranje u GIFT poruke i/ili primanje GIFT poruka (npr. SIMAN poruka) te njihovo prosljeđivanje aplikaciji za poučavanje na način na koji će to aplikacija za poučavanje razumjeti.

1.4. Izmjene i modifikacije u domain modulu

U odgovarajućem trenutku za vrijeme izvršenja GIFT kolegija, modul domene učitava specifičnu datoteku domene koja se zove DKF datoteka (eng. DKF – Domain Knowledge File), koja nije ništa drugo nego XML datoteka čiji je sadržaj ništa drugo do podaci o kolegiju koji opisuju način na koji se kolegij izvršava.

Ova ulazna XML datoteka sadrži određene informacije domene potrebne kako bi domenski modul mogao obavljati nekoliko njegovih ključnih zadataka tijekom učenikove interakcije

s aplikacijom za poučavanje koja se koristi u kolegiju. Prva na redu je procjena uspješnosti učenika na raznim trening zadacima unutar aplikacije za poučavanje. Ona također uključuje i mikro-pedagoška mapiranja učenikovog stanja (afektivno, kognitivno te učenikove performanse) i prijelaza u imenovane nastavne strategije, kao i detalje sprovedbe tih strategija. Integracija svakom novom aplikacijom za poučavanje ili razvijanje novog kolegija pomoću prethodno integrirane aplikacije za poučavanje, obično će zahtijevati stvaranje nove DKF autorske datoteke i to je primarni zadatak u tom slučaju. S tim da u nekim slučajevima, ubacivanje novog Java koda može biti potrebno, ovisno o tome što se želi postići. S obzirom da su DKF datoteke zapravo ništa drugo do XML datoteke, mogu se uređivati s bilo kojim tekst editorom ili XML urednikom, ali najbolja metoda je da se koristi GIFT integrirani autorski alat za oblikovanje područnog znanja (eng. DAT – Domain Authoring Tool). Korištenje DAT-a će provoditi validiranu DKF shemu kao i obavljati druge validacije kao što je provjera protiv vanjskih i nevažećih referenci. Prije stvaranja nove DKF datoteke korisnik treba biti upoznat s formatom DKF datoteke koji je opisan u datoteci `GIFTDomainKnowledgeFile.htm` u mapi sa ostalom GIFT dokumentacijom. Osim toga, svaki od novih GIFT verzija može uključivati jednu ili više testnih dokumenata (proračunskih tablica), od kojih će jedna sadržavati opisni "korak-po-korak" postupak za kreiranje DKF datoteka počevši od nule, kako bi neko drugi tko bude koristio taj kolegij razumio o čemu se radi u DKF datoteci.

Autorska procjena performansi: Autorska procjena performansi se vrši unutar oznaka za ocjenu u DKF datoteci. Osnovna hijerarhijska struktura je

- Zadatak,
- Koncept,
- Stanje

Svaki zadatak ima početne i krajnje okidače te skup pojmova. Svaki pojam, kad dođe njegov red, imat će niz uvjeta. U stvari, primijeti se da je svaki uvjet zapravo oznaka koja sadržava sa `conditionImpl` oznakom koja se odnosi na Java klase odgovorne za obavljanje performansi računanja temeljenih na stanju dobijenom od aplikacije za poučavanje i ulaza kodiranih u DKF datoteci. Trenutne vrijednosti performansi ograničene su na:

- Nepoznato,
- Ispod očekivanja,

- Unutar očekivanja,
- Iznad očekivanja

Osim procjene performansi, svaki uvjet također podržava skup pravila za bodovanje i ocjenjivače koji zajedno određuju konačni rezultat za taj uvjet. Kad su definirana pravila bodovanja, učenicima su prezentirani rezultati pomoću poslije-akcijskog pregleda (eng. AAR – After Action Review) i to nakon njihove izvedbe u odgovarajuće vrijeme sa rezultatima ispisanim na LMS-u. Kao primjer kako i gdje se može koristiti autorska procjena performansi može se navesti sljedeće: kreira se PowerPoint prezentacija sa makroima preko kojih se zadaju zadaci učeniku, te se može implementirati procjena učinka performansi ukoliko se želi znati koliko vremena je učenik potrošio dok je završio pregled PowerPoint prezentacije i riješio zadane zadatke. To se može implementirati na određenim slajdovima ili općenito za cijelu PowerPoint prezentaciju. S druge strane učeniku se mogu poslati odgovarajuće poruke za određene učenikove akcije, npr. ako učenik previše žuri sa pregledom slajdova i samim time ne pročita sve do kraja, ako se previše zadržava na nekom od slajdova bez potrebe i slično. U tim slučajevima učeniku se može preko GIFT sučelja poslati odgovarajuća poruka kako bi uskladio svoje akcije sa lekcijom i što bolje riješio zadatke.

Autorski prijelaz stanja: Autorski prijelaz stanja se obavlja unutar akcijskih oznaka u DKF datoteci. Osnovna struktura je lista prijelaza stanja, od kojih svaki prijelaz predstavlja promjenu stanja kod učenika (ne fizičku, već tijekom korištenja kolegija), na koje učitelj treba reagirati, zajedno s popisom strateških izbora (opcija) koje se mogu koristiti kada se pojavi ta promjena stanja. U slučajevima u kojima se prijelazna stanja odnose na stanja uspješnosti učenikovog učinka, ona će imati referencu natrag do izvedbe čvora u odjeljku za ocjenu unutar DKF datoteke. Npr. u kolegiju kada ima više koncepata koje učenik mora savladati, može se ocijeniti i odrediti procjena učinka za svaki određeni koncept. Odredi se točno gdje koji koncept završava, a gdje počinje, te na taj način rezultat se određuje i upisuje u oznaku za ocjenu u DKF datoteci.

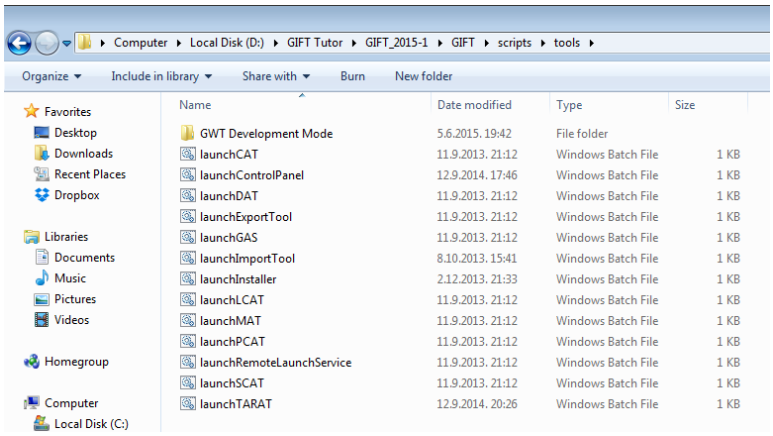
Autorske instrukcijske strategije: Nastavne strategije (eng. Instruction Strategy) također se obavljaju unutar akcijskih oznaka u DKF datoteci. Primijenjene strategije u GIFT-u trenutno uključuju povratne informacije učenika, scenarij prilagodbe (promjene u trenutno izvršavajućem scenariju aplikacija za poučavanje) i zahtjev za ocjenu uspješnosti od strane modula domene. Svaki unos strategija referencira se na `StrategyHandler`, što je

zapravo specifikacija posebne Java klase odgovorne za rukovanje autorskog ulaza sadržanog u DKF datoteci. Povezivanje na Java kod omogućuje znatnu fleksibilnost i to je ono što uveliko olakšava stvar. GIFT se također sastoji i od nekoliko osnovnih modula koji međusobno djeluju kako bi obavljali različite funkcije u CBTS-u. Senzorski modul ima podršku za komercijalne senzore (npr. Affectiva Q-senzor) i njegova funkcija je formatirati, obrađivati i pohranjivati podatke senzora. Domenski modul korisniku postavlja dostupnim sadržaj domene te podržava osposobljavanje, ocjenjuje korisnikovo rješavanje kursa sukladno standardima, te pruža specifične povratne informacije. Trening modul trenira korisnika, koristi povijesne podatke (npr. prošli rad) i podatke senzora za određivanje kognitivnog i afektivnog stanja korisnika. Najnovije izdanje GIFT-a ima funkcionalne elemente svakog od prethodno navedenih modula koji su temeljeni na tutorskoj domeni:

- Modul senzora - uključuje Affectiva Q senzor za mjerenje elektrodermalne aktivnosti (eng. electrodermal activity, EDA), prilagođeni senzor miša za mjerenje elektrodermalne aktivnosti i softverski-temeljen "samoocjenjivački" senzor koji se može manipulirati za testiranje;
- Modul učenika - funkcionira kao prototip prolaza kroz modul, uspoređivanja podataka iz VBS2 (Virtual Battlespace, programska podrška za ratne simulacije bazirane na virtualnoj stvarnosti) i odgovarajućih senzora.
- Pedagoški modul - djeluje na način da daje savjete o konceptima koje je učenik 'slabo' razumio ili dao loš odgovor na njih;
- Domenski modul - sluša VBS2 promet, ocjenjuje izvedbu na 4 različita zadatka, podržava različite razine povratne informacije na zahtjev korisnika;

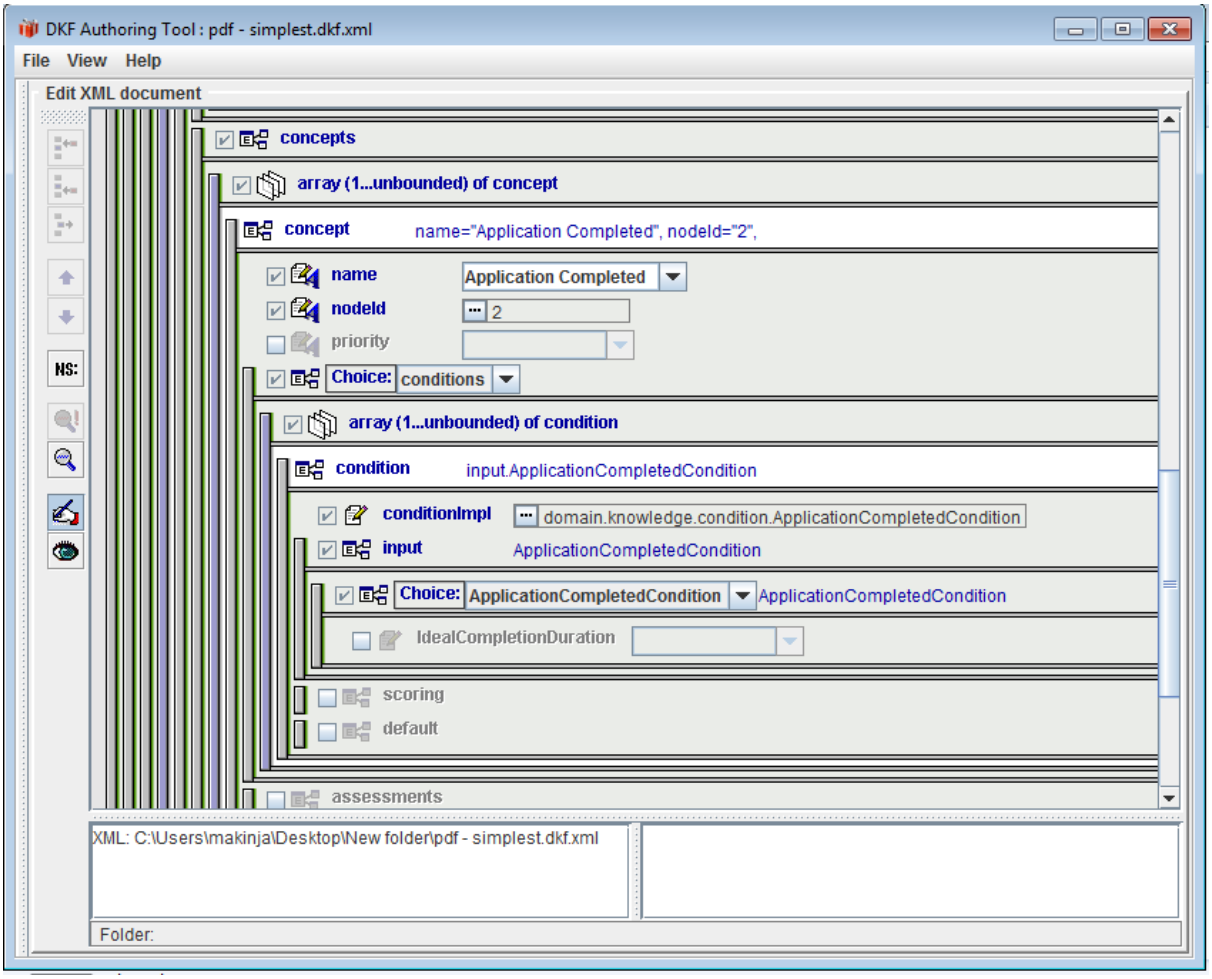
Osim funkcionalnih modula, nekoliko dodatnih komponenti je uključeno za istraživače, uključujući:

- Sposobnost autoru da kreira i koristi vlastita pitanja u istraživanju koristeći se alatom „Survey Authoring Tool“;
- Funkcionalnost praćenja korisnika i njihovih povijesnih izvedbi unutar MySQL baze podataka;
- Sposobnost analiziranja senzora/performansi/procjenjenih podataka nakon sprovedenog eksperimenta preko .csv datoteke, preko vlastitog izbora softvera;



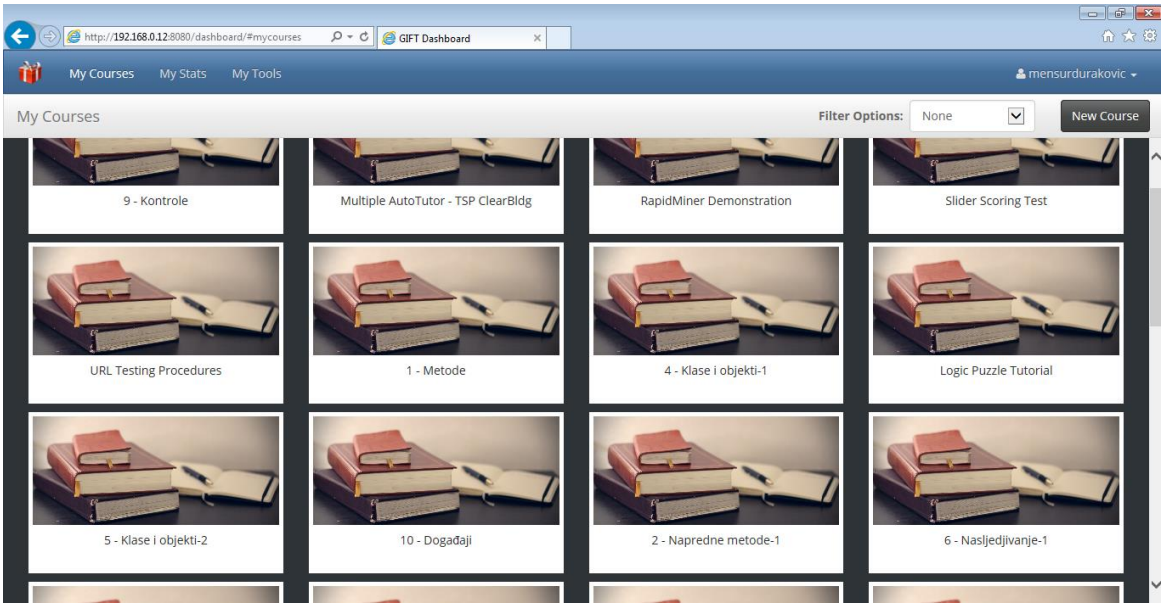
Slika 1 GIFT alati

Primjer jednog od alata je svakako i „DKF Authoring Tool“, koji se koristi za generiranje DKF datoteka koje se mogu koristiti da dobivanje informacija o statistici korištenja određenog kolegija, npr. koliko se vremena korisnik zadržao na nekom slajdu ili prezentaciji, pisanje napomena korisniku tijekom izvođenja kolegija i slično.



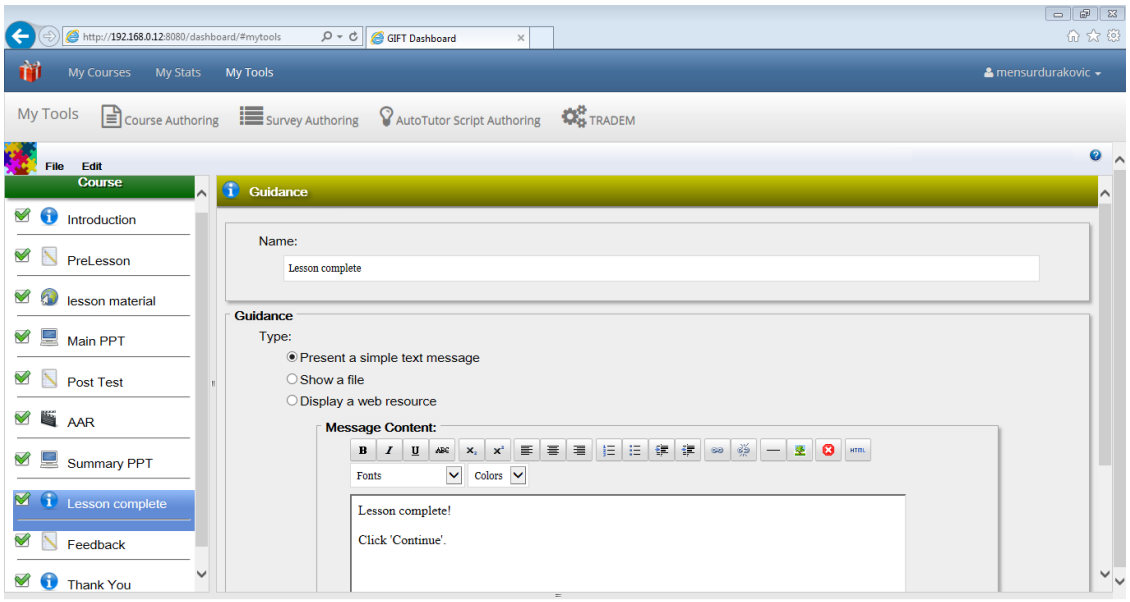
Slika 2 DFK Authoring Tool

S druge strane GIFT-ova glavna kontrolna ploča je alat preko kojeg se može pristupiti svim validiranim kolegijima. Pod validiranim kolegijem se podrazumijeva da je sastavljen u pravilnom formatu i stavke su mu poredane u odgovarajućem redoslijedu po pravilima koje GIFT nalaže.



Slika 3 GIFT Dashboard

Ukoliko se u glavnoj kontrolnoj ploči klikne na dugme „New Course“ otvara se prozor za kreiranje kursa. Kurs se može kreirati i preko CAT alata (Course authoring tool) i sigurno da je bolje koristiti njega u tu svrhu jer se koristeći taj alat odmah može znati da li je kurs u odgovarajućem formatu, složen po pravilima i jako je mala mogućnost pogreške prilikom kreiranja kursa. Ipak, ako je korisnik dovoljno iskusan, može kurs kreirati i preko GIFT-ove glavne kontrolne ploče, s tim da ukoliko kreira nepravilan kurs isti neće moći koristiti dok ga ne popravi.



Slika 4 Kreiranje GIFT kolegija

U ovom radu neće biti objašnjene sve komponente GIFT-a već samo one koje su korištene zbog toga što je naglasak na poučavanju objektno-orijentiranog programiranja. Najčešće korištene komponente su domenski moduli temeljeni na kursevima kroz koje se učeniku objašnjava pojedina nastavna jedinica zajedno sa C# aplikacijom koja služi za utvrđivanje i provjeru znanja.

2. Ispitivanje adaptacije i mogućnosti inteligentnih tutorskih sustava

Inteligentni tutorski sustavi (eng. ITSS – Inteligent Tutoring Systems) kao instrukcijski medij za poučavanje tipa „jedan na jedan“ pokazali su se više učinkoviti u odnosu na tradicionalni sustav učionica koje su usmjerene većinom na frontalni oblik nastave. Također su se pokazali i kao isplativiji, posebno u velikim organizacijama. Procjenjuje se da je multidisciplinskom timu stručnjaka potrebno između 100 i 200 sati utrošenog vremena da kreiraju jedan inteligentni tutorski sustav koji bi isporučio 1 sat nastavnih instrukcija. Pod multidisciplinski tim stručnjaka se mogu svrstati stručnjaci visoko kvalificiranih disciplina poput: računalnih inženjera, dizajnera, psihologa, specijalista za učenje i stručnjaka u drugim domenama.

Potrebno utrošeno vrijeme i vještine autora inteligentnih tutorskih sustava, zajedno s pripadajućom visokom cijenom su glavna prepreka u korištenju, prihvaćanju i usvajanju inteligentnih tutorskih sustava. U ovom poglavlju će biti opisani neki od najvažnijih autorskih izazova i buduće mogućnosti za proizvodnju alata i metoda (automatizacija) u svrhu reduciranja tog mučnog procesa autoriranja tj. kreiranja inteligentnih tutorskih sustava te određene mjere prihvaćanja inteligentnih tutorskih sustava kako bi oni postali uobičajeni za ljudsku zajednicu. Sukladno tom cilju, pregled ovog poglavlja se može podijeliti u dvije kategorije: metode za reduciranje autorskih zahtjeva i metode za automatizaciju autorskih procesa.

Preporuke za budući razvoj alata i standarda unutar GIFT-a su dostupne na internetu i razvrstane su u nekoliko podkategorija:

- „Open-source“ tutorske arhitekture za autoriranje,
- Automatsko upravljanje nastvom,
- Analiza učinka

Razvoj GIFT-a predvodi U.S. Army Research Laboratory s doprinosima od strane akademskih, vladinih i drugih institucija u svojim područjima. U trenutku pisanja ovog diplomskog rada, internet stranica www.gifttutoring.org broji preko 400 korisnika iz 30

zemalja koji isporučuju redoviti tok smjernica i podataka za poboljšanje GIFT izvedbe, kao autorskog, nastavnog i analitičkog alata.

2.1. Autorski ciljevi inteligentnih tutorskih sustava

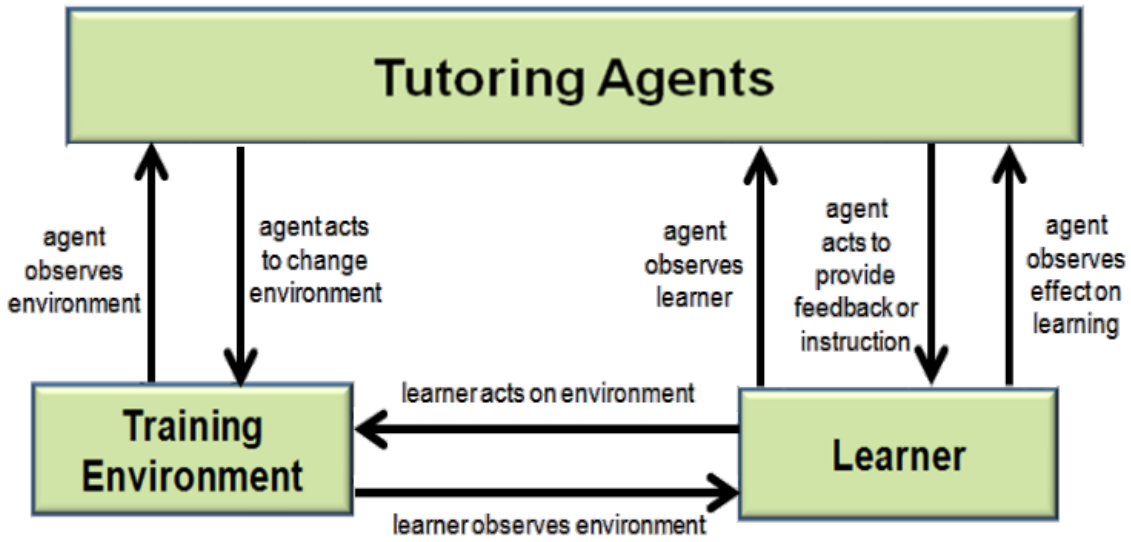
Cilj je smanjiti potrebu za autorom gdje god je moguće kroz standarde interoperabilnosti i ponovnu uporabu. Kako bi se povećao taj cilj, također je potrebno reducirati nužno autorsko znanje i vještine. To se može postići razvijanjem pomagala za posao i drugih elemenata automatizacije koji bi podržali proces autoriranja inteligentnih tutorskih sustava. Cijeli plan sprovođena cilja se može svrstati u nekoliko točaka:

- *Reducirati napor autora* – bilo bi moguće smanjiti napor autora inteligentnih tutorskih sustava uspostavom i dokumentiranjem standarda za postupke, alate i ponovnu uporabu korištenih komponenata. Neke od postojećih komponenata (npr. GIFT, kognitivni autorski alati) mogu biti kandidati za svoje standarde. Predlošci za razvoj modela i sadržaja mogu također smanjiti napor potreban autoru inteligentnih tutorskih sustava. Drugi cilj za lakše autoriranje je uspostaviti standarde za potporu brze integracije vanjskih treninga ili okruženja za podučavanje (npr. video igre, virtualne simulacije, prikaz sadržaja i slično), osigurati ponovno korištenje sadržaja i smanjenje napora autora. Standardi za brzu integraciju vanjskih okruženja će smanjiti potrebu za stvaranjem novih sadržaja za praksu i prezentacije znanja, ali će dodati teret u smislu podudaranja odgovarajućeg sadržaja u odnosu na ciljeve kolegija. Posebna pažnja autori bi trebali pružiti meta-podacima u svrhu lakšeg pretraživanja, lokacije za odgovarajući sadržaj, scenarija za naknadne (buduće) autore.
- *Reducirati donju granicu vještina autora i pomoći u organizaciji znanja* – iako možda ne bi bilo izvedivo da autori imaju potpuno generalizirani set alata za sve discipline, možda je moguće prilagoditi sučelje za autorske alate kako bi se zadovoljile potrebe pojedinih disciplina korisnika (npr. instrukcijski dizajneri, upravitelji kolegija, znanstvenici i stručnjaci u određenom području). Alati za pomoć korisniku u organizaciji znanja za brzo prisjećanje i primjenu mogu rezultirati u velikoj uštedi vremena autoru inteligentnog tutorskog sustava.

- Sprovesti dobre principe dizajna – ovaj cilj nastoji smanjiti broj disciplina i vrijeme potrebno autoru za autoriranje učinkovitog inteligentnog tutorskog sustava. Bitan element dobrog dizajna je sposobnost za podršku (tj. struktura, preporuka, sprovedba) dobrih principa dizajna. To će ublažiti teret autoru na način da će autor znati najbolje primjere pedagoške prakse i optimalne metodologije za interakciju korisničkih sustava.
- Omogućiti brzu procjenu prototipa – ovaj cilj podrazumijeva omogućiti brzu prilagodbu tutorskih sustava kako bi se omogućio brži „dizajn/evaluacija“ ciklus prototipnih sposobnosti. Smanjenje vremena potrebnog za procjenu prototipa će rezultirati u učinkovitijem „model – test - model“ ciklusu i učinkovitijoj podršci autoriranja novih mogućnosti sustava.

2.2. Izazovi za inteligentne tutorske sustave

Adaptivni sustavi, za razliku od prilagodljivih sustava mijenjaju sami sebe kako bi prilagodili interakciju i tako zadovoljili potrebe korisnika. Adaptivni ITS sustavi mijenjaju nastavne strategije (smjer, podršku, povratne informacije), kao odgovor na promjene u statusu stanja učenikovog znanja (kognitivnog, afektivnog i fizičkog) u svrhu poboljšanja njihovog učenja (stjecanje znanja i vještina, zadržavanje itd.).



Slika 5 Interakcija između učenika i ITS-a

Slika 5 pokazuje kako ITS (tutorski agenti i radna okruženja) podržavaju prilagodljive instrukcije o učeniku. Po zoni proksimalnog razvoja, učitelj ima mogućnosti za povećanje/smanjenje razine izazova u okruženju za poučavanje ili povećanje/smanjenje iznosa potpore kako bi uravnotežio učenika između njegovih sposobnosti i težine problemskog prostora kojeg predstavlja okruženje za poučavanje (npr video igra, virtualna simulacija i slično).

Sposobnost inteligentnih tutorskih sustava da se prilagodi potrebama učenika utječe na njihovu cijenu. Više prilagodbenih zahtijeva dodatne sadržaje u domenama koje podržavaju promjene u razini izazova (npr. povećana složenost) u okruženju za poučavanje. Novi sadržaji također mogu doći s uvjetima za dodatne autorske obaveze. Ovaj autorski zadatak općenito je završen prije izvođenja, ali nove tehnologije razvijaju načine da se automatski generira grana alternativnog scenarija. Još jedan izazov inteligentnih tutorskih sustava je sposobnost pomaka s jedne tutorske domene na drugu. Složenost, definicija (dobro ili loše definirano), i dinamika domena koje pokrivaju kognitivne (npr. odlučivanje i rješavanja problema), afektivne (npr. moralni sud, emocionalna inteligencija) i psihomotorne zadatke (npr vještina dobrog gađanja i golf) variraju široko i samim time čine razvoj standarda za modele domena unutar ITS-ova jako velikim izazovom.

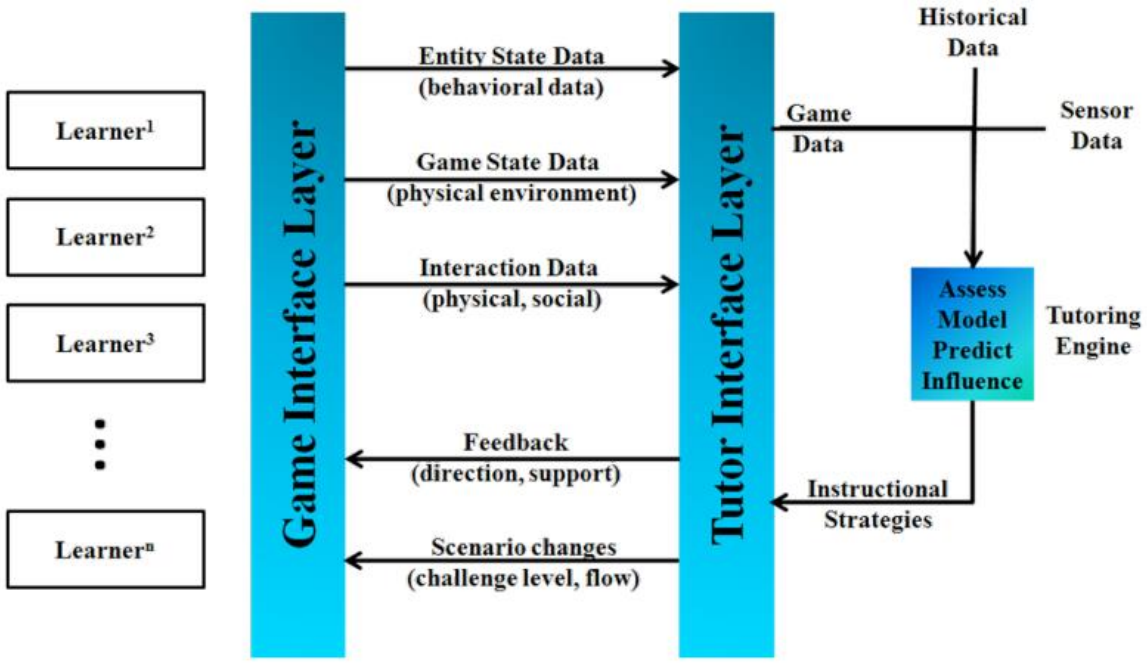
2.3. Metode redukcije autorskih zahtjeva

Ovo poglavlje govori o metodama koje pokazuju obećanje u smanjenju autorskih zahtjeva u inteligentnim tutorskim sustavima. Kao što je objašnjeno ranije, primarni način smanjenja autorskih zahtjeva je promoviranje ponovne uporabe kroz standarde interoperabilnosti i veze na vanjska podučavanje i okruženja za podučavanje (simulacije, igre, alate i ostali sadržaj za poučavanje). U narednim rečenicama će biti opisane tri metode na kojima se trenutno radi, ali još nisu u širokoj uporabi.

2.3.1. Interoperabilnost s ozbiljnijim video igrama

Gateway modul unutar GIFT-a omogućuje standard interakcija za integraciju vanjskih tutorskih sustava i okruženja za podučavanje. Popularnost ozbiljnih igara koje su zapravo virtualne igre koriste se za obuku taktičkih zadataka (npr zemljišna navigacija) i ono donosi element visokog angažmana za poučavanje temeljeno na igrama. Sustavi za

poučavanje temeljeni na igrama koriste kvalitete igre zajedno sa najboljim nastavnim praksama tutora kako bi pružili korisnicima adaptivna „jedan-na-jedan“ iskustva u učenju. Scenariji ozbiljnijih video igara mogu se ponovno koristiti, kopirati i brzo modificirati u uredniku igračkog scenarija te na taj način podržati složenije grananje za potrebe prilagodbe.



Slika 6 Poučavanje temeljeno na igrama

Slika 6 prikazuje vrstu razmjene podataka između sloja sučelja igre i sloja za sučelje tutora. Stanje entiteta, statistike igre i interakcijski podaci koje generirani od učenika tvore glavni element odluke u inteligentnim tutorskim sustavima, dok su povratne informacije i strategije promjena scenarija preporučene od strane mentora implementirani kao taktike u igri. U tijeku su procjene za vrednovanje učinka ozbiljnih igara s GIFT sustavom i AutoTutor sustavom. Do danas, GIFT je integriran s Virtual BattleSpace2 (VBS2), vMedic i Unity Game Engine sustavima. Standardizacija razmjene podataka između igre i sloja sučelja tutora je u neposrednoj blizini i daje priliku za povećanje interoperabilnosti, smanjenje vremena za autoriranje i integraciju jedinstvenog „učitelj-igra“ odnosa uparivanja. O automatizaciji ovog procesa će biti riječi u poglavlju "Metode za automatizaciju autorskih procesa" ovog rada.

2.3.2. Interoperabilnost s vanjskim web servisima

U sklopu posljednjih verzija GIFT-a, ARL provodi pozive na vanjskim AutoTutor web servisima. Web servisi dostupni putem GIFT-a podržavaju AutoTutor dijaloški temeljeno poučavanje uključuju sljedeće elemente: latentna semantička analiza (LSA) teksta podržava analizu učenikovih odgovora gotovo u stvarnom vremenu; razgovorni dijalozi na temelju procjene LSA; sučelje za animirane agenata te razne druge tutorske i stilske mehanizme isporuke. Pozivi web servisa su podatkovno pogonjeni i stoga uglavnom domenski neovisni. Sučelja su standardizirani s GIFT-om za podršku interakcije s komercijalnim virtualnim učesnicima putem „Media Semantic“ skupa znakova. Ovaj dodatak GIFT-a smanjuje potrebu za programiranje pojedinačnih interakcija između tutora i učenika, a time se zapravo smanjuje i opterećenje tutora (autora).

2.3.3. Interoperabilnost s hardverom i softverski temeljenim metodama prikupljanja podataka

Još jedna značajka GIFT-a je i ponovna uporaba hardverskih sučelja. U nekim slučajevima, može biti potrebno prikupiti podatke o učeniku za podršku procesa poučavanja u realnom vremenu. Ti podaci mogu biti prikupljeni kroz akcije učenika (npr. govoreći, tipkajući) i softverski-temeljenim metodama snimanja. ARL je također stvorio softverski senzor, koji se može koristiti za ispitivanje osjetljivosti i eksperimentiranje u odsutnosti drugih senzora. Podaci mogu biti „zarobljeni“ od strane temelji hardverskih senzora. Od inicijalnog izdanja, ARL je integrirao nekoliko hardverski temeljenih senzora kao dio svog eksperimentalnog razvoja. Ovi senzori trenutno uključuju sučelja za „Emotive epoch“ senzore (trgovački jeftine varijante elektroencefalograma (EEG)), afektivne Q-senzore (komercijalni elektro-dermalni senzor korišten za mjerenje različitih aktivnosti), Microsoft Kinect senzore (trgovački senzor sa softverski-temeljenim podsustavima za detekciju djelovanja i otkrivanje fizičkog stanja) i niz fizioloških senzora.

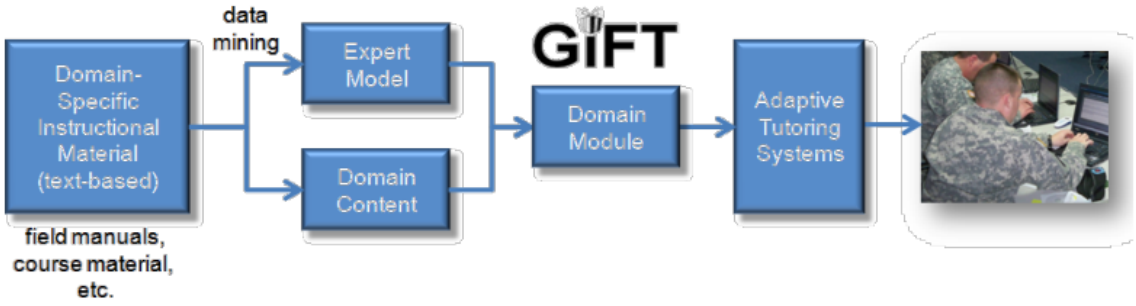
2.4. Metode automatizacije autorskih zahtjeva

Ovo poglavlje govori o metodama koje pokazuju obećavajuće rezultate kod automatizacije autorskih procesa u inteligentnim tutorskim sustavima. Automatizacijom procesa, možemo smanjiti opterećenje autora i potrebno znanje autora koje je neophodno u određenoj mjeri

kod procesa autoriranja inteligentnih tutorskih sustava. Konkretno za GIFT, cilj je biti u mogućnosti pružiti alate prikladne za korisnike poput ne-računalnih znanstvenika koji su stručnjaci u drugim domenama, sposobnost autoriranja inteligentnih tutorskih sustava u svrhu podržavanja izrade lekcija iskoristivih u raznim domenama (kognitivna, afektivna, psihomotorna, socijalna, hibridna) koje su trenutno potrebne u preko 50% slučajeva za korisnike koji imaju status početnika. Ispod su opisane dvije metode koje su se pokazale kao dobre, ali koje još nisu u širokoj uporabi.

2.4.1. Automatizacija razvoja stručnih modela

Koncept automatskog razvoja stručnih modela obično se referira kao TRADEM (eng. Tools for Rapid Automated Development of Expert Models) i nudi smanjene vremena i vještina, a time i troška, za razvoj bitnih dijelova trening domena bez ljudskog znanja o konkretnoj domeni. ARL (eng. Army Research Laboratory) trenutno surađuje s kompanijom „Eduworks“ (mala tvrtka u Oregonu), kako bi razvili alate i metode za automatizaciju razvoja stručnih modela korištenih u GIFT-u u svrhu proizvodnje adaptivnih tutorskih alata. To nastoje postići rudarenjem podataka specifičnih nastavnih materijala pomoću tehnika za izdvajanje pravila, načela, zadataka, standarda, uvjeta i hijerarhijskih odnosa s tekstom u poljima za uporabu (Slika 7). Stručni modeli, koji su dio GIFT domenskog modula, koriste se za procjenu učeničkih performansi i ispravnost učenikovih akcija tijekom sesija poučavanja.



Slika 7 – TRADEM proces

2.4.2. Automatizacija razvoja posrednog softvera za integraciju igara i tutora

Kao što je spomenuto u poglavlju "Metode za smanjenje autorskih zahtjeva" ovog rada, mogućnost automatizacije integracije igara i tutorskih sustava će rezultirati kombiniranjem

viših razina angažmana koji se mogu pronaći u kompleksnijim video igrama, zajedno s učinkovitijim tehnikama učenja pronađenim u tutorskim sustavima. ARL surađuje s tvrtkom „CHI Systems“ (Pennsylvanija), kako bi razvili alate za automatizaciju procesa razvijanja posrednog softvera koji će povezati kompleksnije video igre i inteligentne tutorske sustave. Ovaj razvojni alat posrednog softvera je obično poznat kao GAMETE (eng. Game-based Architecture for Mentor-Enhanced Training Environments) i taj proces će biti integriran u GIFT u budućim verzijama.

2.5. Zaključci i preporuke za buduća istraživanja

U ovom poglavlju identificirane su mogućnosti za reduciranje autorskog napora putem standardizacije i interoperabilnosti u svrhu podrške povećanja ponovne uporabe njegovih dijelova i modela, kao i smanjiti vrijeme i vještine potrebne autoru inteligentnih tutorskih sustava kroz proces automatizacije. Kroz GIFT i njegovu korisničku zajednicu, ARL pokušava standardizirati komponente i sučelja za promicanje poboljšane interoperabilnosti. ARL također provodi istraživanja na generalizaciji alata i metoda kako bi u GIFT implementirali podršku za smanjeno vrijeme/vještine kod kreiranja inteligentnih tutorskih sustava koji podržavaju kognitivne, afektivne, psihomotorne, socijalne i hibridne tutorske/trening domene. GIFT također pruža ozbiljan alata i predložaka za ublažavanje autorskog napora i tereta. Konačno, GIFT je projektiran na način da podržava spajanje generaliziranih usluga, što može biti na raspolaganju raznim video igrama i simulacijama za podršku samoregulirajućeg učenja na zahtjev. Dodatna istraživanja su potrebna nad sljedećim elementima kako bi se utvrdile optimalne metode za kvalitetnije autoriranje unutar GIFT-a:

- Tehnike, strategije i taktike unutar GIFT-a,
- Procjene učinka i kompetentnosti unutar GIFT-a,
- Sadržaj domena i razvoj stručnih modela,
- Novi sadržaj i nastavni modeli iz postojećih izvora podataka

3. Analiza GIFT korisničkih iskustava i korištenja

Tijekom protekle četiri godine, programeri i projektanti GIFT-a su usmjerili svoje napore da istraživanjima u nastavi olakšaju što je više moguće, pa su se odlučili na kreiranje inteligentnih tutorskih sustava (eng. ITS – Inteligent Tutoring System) i sve to u svrhu toga kako bi istraživačima omogućili što lakše ocjenjivanje učinkovitosti učenja u domenskom modulu na neovisan način. GIFT programeri su zajedno sproveli veliki broj sastanaka, sjednica i proučili čitav niz knjiga kako bi identificirali ključne izazove koji se odnose na inteligentne tutorske sustave te kako dati preporuke za dizajn na način na koji GIFT to može riješiti. Dodatna svrha tih napora je bilo i povećanje općeprihvaćenosti GIFT-a među korisničkom zajednicom te razumijevanje bitnih značajki i funkcionalnosti neophodnih za potporu istraživanjima.

Iako je GIFT trenutno još uvijek u fazi prototipa za istraživačke radove, povećao je broj aktivnih korisnika koji sada iznosi preko 300 korisnika. Tako da, jedna od najvećih zadaća GIFT-a je povećati GIFT iskoristivost i poboljšati interakcije u smislu dizajna a sve u cilju promicanja pozitivnog korisničkog iskustva. Da bi neka interaktivna tehnologija (pa tako i GIFT-a) bila uistinu učinkovita i ugodna iz perspektive korisnika, ona mora biti dizajnirana za upotrebljivost i ugodno korisničko iskustvo. U ovom radu će biti predstavljeni rezultati ankete sprovedene na GIFT korisnicima koja se temelji na njihovim percepcijama upotrebljivosti GIFT-a i njihovih korisničkih iskustava.

3.1. Upotrebljivost/korisničko iskustvo

Na samom početku se postavlja pitanje, koja je razlika između upotrebljivosti i dobrog korisničkog iskustva? Uloga upotrebljivosti nekog programa je osigurati da interaktivni program poput GIFT-a bude efektivan, učinkovit i siguran za korištenje, da posjeduje dobre značajke i funkcionalnosti, da je lagan za učenje i korisniku pruža lako pamtljiv način korištenja. Dobra upotrebljivost zapravo znači da korisnik može uspješno ostvariti svoje ciljeve/zadatke koristeći interaktivni program u kojemu su implementirane prethodno navedene karakteristike.

S druge strane, ugodno korisničko iskustvo se odnosi na koncept korisnikovog osjećaja tijekom korištenja sustava. Iskustvo korisnika se gleda više u smislu subjektivne prirode i može uključivati elemente za procjenu u kojima korisnik percipira tehnologiju kao zadovoljavajuću, ugodnu, zanimljivu, motivirajuću, estetski ugodnu, stimulirajuću s kognitivnog pogleda i slično. Negativna percepcija korisničkog iskustva se može opisati sa frustruacijama, dosadom, nezadovoljstvom prilikom korištenja tehnologije i sličnim elementima.

Procjena o tome što se gradi tijekom cijelog procesa i korisničkog iskustva kojeg proces nudi su temeljni elementi u kreiranju tehnologije i interakcijskog dizajna. Npr. računala i video igre su ostvarili veliki uspjeh na razumijevanju važnosti tih odnosa projektirajući tehnologije za oba elementa, upotrebljivosti i korisničko iskustvo. To je vidljivo i po kontinuiranom rastu njihovih prihoda (u milijardama) godišnje.

3.2. Projektiranje za upotrebljivost i korisničko iskustvo

Iako projektiranje za korisičko iskustvo može biti više izazovno u odnosu na projektiranje za upotrebljivost, dizajniranje za oba elementa će proizvesti maksimalno korisničko prihvaćanje i korištenje. Postoji više vrsta načela dizajna interakcijskih tehnologija poput GIFT-a i neki od njih su razvijeni od strane stručnjaka u području interakcije čovjeka i računala:

- Don Norman je napisao djelo „Principi dizajna“ u kojem opisuje skup načela koja uključuju vidljivost, povratnu informaciju, ograničenja, mapiranje, dosljednost i priuštivost.
- Jakob Nielsen autor je djela „10 heuristika upotrebljivosti za dizajn korisničkog sučelja“ koji prema njemu uključuju vidljivost statusa sustava, usklađenost između sustava i stvarnog svijeta, korisničku kontrolu i slobodu, konzistentnost i standarde, prevencija pogrešaka, prepoznavanje umjesto opoziva, fleksibilnost i učinkovitost uporabe, estetski i minimalistički dizajn, pomoć korisnicima pri prepoznavanju, dijagnosticiranju i oporavljanju od pogrešaka te pomoć i dokumentacija.
- Ben Shniedermanovo djelo „Osam zlatnih pravila sučelja dizajna“ u kojem uključuje težnju dosljednosti, omogućavanje korisnicima korištenje prečaca, ponudu povratne informacije, ugodan dizajn dijaloga, jednostavno rukovanje

pogreškama i obrat radnji korisnika, podržavanje unutarnje kontrole korisnika, smanjenje opterećenja kratkoročne memorije itd.

Iz gore navedene liste može se zaključiti da su vidljivost, dosljednost i povratna informacija najčešće ponovljeni koncepti i ujedno i najučestaliji. Ipak, ta načela mogu biti korisna za projektiranje, razvoj i vrednovanje aspekata u kreiranju bilo koje interaktivne tehnologije pored GIFT-a.

Nažalost, temeljito istraživanje upotrebljivosti i korisničkog iskustva traži puno vremena. Jedan od bržih načina za razumijevanje percepcije korisnika prema interaktivnoj tehnologiji je da se pregledaju i uključe elementi tehnološki prihvatljivih modela (eng. TAM - Technology Acceptance Model). TAM je teorijski model koji predviđa kako će korisnik prihvatiti i koristiti određenu tehnologiju. TAM model navodi jednostavne odnose vanjskih varijabli, uvjerenja, stavova i stvarnog ponašanja korisnika. Model sugerira da kada su korisnici prezentirani određenom informacijskom tehnologijom tj. brojnim čimbenicima posebno procjenom korisnosti (eng. PU – Perceived usefulness) i procjenom lakoće korištenja (eng. PEU – Perceived ease of use), da to utječe na njihovu odluku o tome kako i kada će koristiti tehnologiju. Izvorni TAM model ocjenjuje korsnikove kognitivne, afektivne i bihevioralne odgovore na pitanje o određenoj tehnologiji. Elementi procjene korisnosti i procjene lakoće korištenja predstavljaju korisnikove kognitivne odgovore na korištenje tehnologije. Ti kognitivni odgovori se prenose dalje na korisnikove afektivne (stav prema korištenju) i bihevioralne (buduće namjere korištenja) odgovore na korištenje tehnologije. U ovom radu će biti objašnjena anketa koja koristi samo elemente procjene korisnosti i procjene lakoće korištenja te budućim namjerama korisnika korištenja GIFT-a.

3.3. Metodologija

Anketa zadovoljstva korištenja GIFT-a je kreirana i poslana preko službenog GIFT e-mail servisa. Od preko 300 GIFT korisnika registriranih preko www.gifttutoring.org web stranice, 8 korisnika je odgovorilo na anketu. Važno je spomenuti kako GIFT programeri i dizajneri nisu sudjelovali u anketi kako bi se osigurala nepristranost rezultata. 4 korisnika koji su odgovorili na anketu su bili istraživači/analitičari, 2 korisnika su bila ITS autori/treneri, 1 korisnik je bio instruktor/trener i 1 korisnik je bio programer. Ispitanici su identificirali sljedeće svrhe za korištenje GIFT-a (moglo se izabrati više odgovora): sedam

odgovora je zabilježeno za korištenje GIFT-a u svrhu istraživanja i razvoja, tri odgovora su bila za odabir GIFT kao alata za eksperimentiranje te dva odgovora za GIFT kao alat za vrednovanje svojih sposobnosti. Pitanja u anketi su bila podijeljena po sljedećim sekcijama:

1. Instalacija GIFT-a, dokumentacija i podrška (4 pitanja),
2. Procjena lakoće korištenja GIFT-a (11 pitanja),
3. Procjena upotrebljivosti GIFT-a (5 pitanja),
4. Buduće korištenja GIFT-a (2 pitanja),
5. Potencijalne značajke i funkcionalnosti koje bi se mogle dodati u GIFT tj. preporuke za poboljšanje GIFT-a (5 pitanja)

Rezultati ankete za svaku sekciju su prikazani u sljedećem poglavlju.

3.4. Rezultati ankete i diskusije

3.4.1. Instalacija GIFT-a, dokumentacija i podrška

Tablica 1 prikazuje rezultate korisničkih percepcija na instalaciju GIFT-a, dokumentaciju i podršku. Pitanja u ovoj sekciji su temeljena na 6-stupanjskoj Likertovoj ljestvici gdje broj 1 označava termin „jako nezadovoljni“ i broj 6 termin „jako zadovoljni“.

#	Izjava	Min, Max	Prosjek	Stand.devijacija
1	Jednostavnost instalacije GIFT-a	1,6	4.00	1.852
2	Potpunost i točnost uputa za instalaciju	1,6	4.13	1.553
3	Potpunost i korisnost GIFT dokumentacije	1,5	3.63	1.408
4	Dostupnost tehničke podrške (gifttutoring.org forumi itd.)	4,6	4.88	0.991

Ovi rezultati sugeriraju da GIFT korisnika uglavnom smatraju GIFT jednostavnim za instalaciju (M = 4.0, SD = 1,852). Uglavnom su zadovoljni sa potpunošću i točnošću GIFT uputa (M = 4.13, SD = 1,553) kao i potpunošću i korisnošću GIFT dokumentacije (M =

3.63, SD = 1.408). Zadovoljni su i sa dostupnošću GIFT tehničke podrške (M = 4.88, SD = 0,991).

3.4.2. Procjena lakoće korištenja GIFT-a (PEU)

Tablica 2 prikazuje rezultate korisničke procjene lakoće korištenja GIFT-a (eng. PEU – Perception of Usage). PEU označava termin koji definira stupanj do kojeg korisnik smatra da će korištenje određenog sustava biti moguć bez ulaganja napora. Pitanja korištena u ovoj sekciji se temelje na 6-stupanjskoj Likertovoj ljestvici gdje broj 1 označava termin „uopće se ne slažem“, dok broj 6 označava termin, u potpunosti se slažem“.

#	Izjava	Min, Max	Prosjek	Stand.devijacija
1	Moja interakcija s GIFT-om je jasna i razumljiva	1,4	2.63	1.188
2	Interakcija s GIFT-om ne zahtijeva puno mentalnog napora	1,5	2.75	1.488
3	Smatram da je GIFT lagan za korištenje	1,5	2.38	1.302
4	Mislim da je u GIFT-u lako napraviti ono što želim	1,4	2.63	1.061
5	Imam kontrolu nad korištenjem GIFT-a	1,4	3.25	1.165
6	Smatram korištenje GIFT-a ugodnim	1,4	2.88	0.991
7	Smatram GIFT fleksibilnim alatom po pitanju interakcije	1,4	2.25	1.035
8	Lako sam naučio/la obavljati zadatke koristeći GIFT	1,5	1.75	1.488
9	GIFT ima dobru funkcionalnost i značajke	4,5	4.50	0.535
10	Smatram da imam dobru intuiciju kako upravljati GIFT-om	1,6	2.88	1.642

11	Smatram da je lako zapamtiti kako obavljati zadatke koristeći se GIFT-om	1,5	3.63	1.302

Primarni rezultati upućuju na to da korisnici misle kako GIFT ima dobru funkcionalnost i značajke (M = 4.50, SD = 0.535) i da je lako pamtljivo kako raditi zadatke u njemu (M = 3.63, SD = 1.302). Međutim, korisnici ne osjećaju kako je GIFT lagan za korištenje. Dok god je ovo slučaj, ovi rezultati mogu poslužiti kao nekakva osnovica za buduća istraživanja o GIFT-u. Preporuka za budući razvoj GIFT-a na temelju ovih rezultata bi bila povećanje fleksibilnosti u smislu autorskih mogućnosti kojima GIFT raspolaže.

3.4.3. Procjena upotrebljivosti GIFT-a (PU)

Tablica 3 prikazuje korisničku procjenu upotrebljivosti (eng. PU – Perceived usefulness) GIFT-a. PU se može definirati kao stupanj do kojeg korisnik smatra da bi korištenje određenog sustava unaprijedilo ili poboljšalo njegove poslovne izvedbe. Pitanja u ovoj sekciji se temelje na 6-stupanjskoj Likertovoj ljestvici gdje broj 1 označava termin „uopće se ne slažem“ i broj 6 termin „u potpunosti se slažem“.

#	Izjava	Min, Max	Prosjek	Stand.devijacija
1	Korištenje GIFT-a poboljšava moju uspješnost u obavljanju posla	1,4	2.63	1.188
2	Korištenje GIFT-a poboljšava moju produktivnost	1,4	2.50	1.195
3	Korištenje GIFT-a poboljšava moju učinkovitost u obavljanju posla	1,4	2.75	1.035
4	Smatram GIFT korisnim sustavom za moj posao	1,5	3.13	1.356
5	Za moj posao, korištenje GIFT-a je relevantno	2,6	4.38	1.302

Na temelju ovih rezultata ispitanici ne osjećaju da GIFT poboljšava izvedbu u njihovom poslu (M = 2.63, SD = 1.188), niti da povećava njihovu produktivnost (M = 2.50, SD = 1.195) ili da poboljšava učinkovitost njihovog posla (M = 2.75, SD = 1.035). Međutim, slažu se da je GIFT relevantna tehnologija u njihovom poslu (M = 4.38, SD = 1.302). U biti, procjena upotrebljivosti GIFT-a među ispitanicima je trenutno niska. To i ne čudi iz dva razloga: Prvi razlog je taj da je PEU primarni element koji doprinosi PU u TAM-u i drugi razlog je taj da razina PU-a ovisi o nužnosti tehnologije i na taj način direktno utječe na produktivnost posla. Npr. e-mail aplikacije će proizvoditi visoke razine PU-a jer su jako isprepletene u komunikaciji među korisnicima. Kada se ponovo pogledaju kategorije poslova ispitanika GIFT-a te njihove glavne namjere za korištenje GIFT-a očito je da su uloge i odgovornosti tih kategorija ogromne i da uporaba GIFT-a među njima predstavlja jako mali dio. Kako se PEU povećava tako će rasti i PU, posebno iz razloga što ispitanici smatraju da je GIFT relevantna tehnologija za njihov posao. Jedna preporuka koja proizilazi iz gore spomenutih analiza je da bi trebalo povećati GIFT-ovu procjenu upotrebljivosti i to na način da se istaknu i prošire značajke i funkcionalnosti GIFT-a bazirane na ulogama različitim vrsta korisnika.

3.4.4. Buduće korištenja GIFT-a

Tablica 4 prikazuje rezultate korisničke vjerojatnosti budućeg korištenja GIFT-a. Pitanja u ovoj sekciji se temelje na 6-stupanjskoj Likertovoj ljestvici gdje broj 1 označava termin „malo vjerojatno“ dok broj 6 označava termin „vrlo vjerojatno“.

#	Izjava	Min, Max	Prosjek	Stand.devijacija
1	Koja je vjerojatnost da ćete koristiti GIFT u budućnosti za svrhe koje ste prethodno utvrdili ?	3,6	4.63	1.126
2	Koja je vjerojatnost da ćete Vi doprinijeti budućnosti GIFT-a širenjem njegovog izvornog koda i vlastitim otkrićima korisničkoj zajednici GIFT-a ?	3,6	5.13	0.916

Ovi rezultati ukazuju na to da će ispitanici vrlo vjerojatno nastaviti s korištenjem GIFT-a u budućnosti (M = 5.13, SD = 1.126) te da će vrlo vjerojatno doprinijeti budućem širenju GIFT izvornog koda sa dijeljenjem vlastitih analiza među korisničkom zajednicom GIFT-a (M = 4.63, SD = 0.916). To je jako važno iz razloga što pokazuje da korisnici cijene vrijednost i motivaciju koja stoji iza razvoja GIFT-a. Ovo također podržava prethodnu analizu u kojoj je GIFT definiran kao relevantna tehnologija u poslovima ispitanika.

3.4.5. Dodatne značajke i potrebne funkcionalnosti

Ova sekcija se sastoji od 5 pitanja otvorenog tipa. U nastavku su pitanja i odgovori za svako od njih.

1) Koje autorske značajke i funkcionalnosti trebate u svom ITS okruženju a koje GIFT ne nudi?
    a) Interakcije na prirodnijem jeziku,
    b) Brze iteracijske značajke (autorske ankete, autorski kolegiji),
    c) Način za poučavanje tima ljudi,
    d) Više logički orijentiran sustav koji nije temeljen na XML datotekama, nešto što je više bazirano na dijagramima toka.
2) Koje instrukcijske značajke i funkcionalnosti trebate u svom ITS okruženju a koje GIFT ne nudi?
    a) Intergracija s LMS sustavima,
    b) Sposobnost prioriteta koja instrukcijska strategijaje odabrana za sanaciju ukoliko je više uvjeta istodobno ispod očekivanja,
    c) U XML stablastim strukturama je lako izgubiti orijentaciju pa je teško odlučiti što se popravlja li dodaje,
    d) Nisam dovoljno dugo koristio GIFT da kažem nešto o ovoj temi/Ništa/Nisam siguran.
3) Koje analitičke značajke i funkcionalnosti trebate u svom ITS okruženju a koje GIFT ne nudi?

a) Za očekivano vrijeme završetka određene aktivnosti bilo bi lijepo imati niz ili razdiobu vjerojatnosti. Također, dobro bi došlo više granulacije prema izvedbama poput autorski definiranim izvedbama, koristeći sofisticiranija pravila,

b) Spajanje više podatkovnih kanala u skup podataka (npr. podaci senzora, podaci logova), istraživanja alata za podatke,

c) Sposobnost analize različitih članova tima istovremeno,

d) Nisam dovoljno dugo koristio GIFT da kažem nešto o ovoj temi/Ništa/Nisam siguran.

4) Navedite barem 1 prijedlog kako GIFT napraviti više korisnički orijentiranim i lakšim za korištenje.

a) Napraviti demo video ili set video datoteka koje bi opisale svaku od značajki/funkcionalnosti.

b) XML stablo treba da se ispolira. Tema svakog od ogranaka ili „roditelja“ može biti istaknuta sa većim fontom ili nečim drugim. Validacija stabla nije u realnom vremenu. Potrebno je sačuvati XML datoteku, validirati je, zatim sačuvati ponovo. Na nekim mjestima dugmad nije lako za pronaći.

c) Instalacijski čarobnjak sa opcijama za konfiguraciju GIFT-a. Svesti na najmanju moguću mjeru ažuriranje XML datoteka tijekom instalacije. Također, riješiti problem ovisnosti o drugim programima (npr. MySQL, PowerPoint, itd.). Ukoliko neki od osnovnih programa nedostaje, to ne bi trebao sprječavati pokretanje GIFT-a. Također, bilo bi korisno sprovesti heurističku procjenu na autorske ankete i autorske kolegije. Sa svim tim značajkama, potrebno je previše klikova mišem da bi se stvari uspostavile,

d) Dokumentacija se može malo dopuniti i ažurirati u svrhu pomaganja novim korisnicima,

e) Proširiti tehnički dio dokumentacije kako bi više koristila inžinjerima/sistemcima (npr. korištenje XML konfiguracijskih datoteka, očekivani redoslijed za generiranje poruka i privitaka rijekom obrade).

5) Postoje li neke značajke ili funkcionalnosti koje bi trebale biti integrirane u GIFT ?

a) Dodatne API funkcije i pozivi za okolne sustave GIFT poruka, npr. sposobnost dodavanj novih vrsta GIFT poruka i postaviti pravila u sustavu kako bi GIFT trebao reagirati na njih.

b) Robusniji domenski logički API tako da bi i druge simulacije mogle slati GIFT poruke o svojim podacima bez da stvaraju nove uvjete. I možda jasnija dokumentacija kako bi to sve trebalo raditi u kompleksnijim slučajevima.

c) Prethodne povratne informacije su temeljene na GIFT verziji 3.0 u odnosu na novu verziju 4.0. Pored gore napisanih kritika, veoma sam oduševljen/a sa GIFT-om, no mislim da postoji puno mogućnosti za poboljšanje korisničkog iskustva.

d) Nisam dovoljno dugo koristio GIFT da kažem nešto o ovoj temi/Ništa/Nisam siguran.

Ovih 5 pitanja otvorenog tipa pružaju kvalitativni aspekt zadovoljstva ispitanika ankete. Odgovori na ova pitanja daju uvid u načine povećanja upotrebljivosti i korisničkog iskustva. Također pomažu pri objašnjenju prije navedenih kvantitativnih rezultata ankete.

3.5. Zaključak

Sveukupno gledajući, ispitanici ankete izrazili su pozitivne stavove prema GIFT-u u smislu buduće namjere korištenja, njegove instalacije, dokumentacije i podrške. Međutim, nisko su ocijenili jednostavnost uporabe i korisnost GIFT-a, osim nekoliko značajnih pitanja o kojima se raspravljalo u analizama. Ipak, to otkriće će koristiti kao osnovica za buduća istraživanja o zadovoljstvu. Ispitanici cijene GIFT i ono čemu on teži, sada i u budućnosti. Dobra vijest je da su sve željene buduće promjene u rezultatima ankete u planu za implementaciju u skorijoj budućnosti. Neke od zahtijevanih promjena poput više korisnički orijentiranog sučelja za kreiranje autorskih kolegija će biti implementirane u sljedećoj verziji GIFT-a. Kako GIFT programeri budu radili na GIFT-ovoj korisnosti i optimizaciji za korisničko iskustvo, buduće predikcije će biti predstavljene GIFT korisničkoj zajednici kako bi se osiguralo maksimalno korištenje i prihvaćanje.

4. Objektno-orijentirano programiranje

Objektno-orijentirano programiranje (eng. Object-Oriented Programming, OOP) je način pristupa realizaciji softvera kao modelu realnog svijeta. OOP predstavlja sasvim nov način razmišljanja u odnosu na tradicionalno algoritamsko programiranje. Problemi se identificiraju kao objekti koji nešto rade i apstrakcije koje predstavljaju skupove objekata istih svojstava. Kod ovakvog načina programiranja puno više vremena se troši na projektiranje, a manje na samo programiranje, što je bio obrnut slučaj kod običnog (tradicionalnog) programiranja. OOP u centar rada postavlja problem koji rješava, a ne pitanje konkretnog programskog rešenja. Naglasak je na objektima koji nešto rade, a ne na algoritmima tj. na tome kako nešto radi. OOP pruža nekoliko bitnih koncepata, a to su: klase, objekti, enkapsulacija, metode i svojstva, konstruktori, nasljeđivanje i polimorfizam.

U ovom radu, s razlogom je odabrano OOP kao glavna tema iz razloga što veliki broj studenata ima problema sa shvaćanjem osnovnih koncepata OOP-a. Kroz ovaj rad pokušao sam sve svoje probleme i greške koje sam radio dok sam učio osnove OOP-a pretočiti i pokušati korisnika aplikacije za poučavanje OOP-a uputiti na pravo razmišljanje. Najveći problem koji se danas javlja je nedostatak primjera u neposrednoj praksi i primjeni u praksi. Većina nastavnika se usredotoči na objašnjavanje koncepata i definicije istih, dok se kasnije nastavlja sa navođenjem primjera. Odabir primjera za pokazati nekome po prvi put određeni koncept OOP-a je od ključne važnosti. Taj primjer mora biti logičan, jasan, kratak i po mogućnosti da to bude nešto iz svakodnevnog života, lako razumljivo i jednostavno. Ovaj rad obiluje primjerima i jednostavnim objašnjenjima jer je glavni fokus na tome.

4.1. Poučavanje objektno-orijentiranog programiranja

Razvojna zajednica se više ili manje slaže da je poučavanje objektno orijentiranog programiranja u nastavi odlična stvar. Objektno orijentirano programiranje na elegantan način pruža podršku za koncepte koji se pokušavaju učiti dugi niz godina, kao što je strukturirano programiranje, modularizacija i programski dizajn. Ono također podržava

tehnike za uočene probleme koji su tek nedavno pronašli svoj put u kurikulumima: programiranje u timovima, održavanje velikih sustava i „reuse“ ili iskoristivost softvera. Ukratko, objektno orijentirano programiranje se čini kao solidan alat za učenje tih programskih metodologija koje se danas smatraju važnima. Nastava objektno orijentiranog programiranja, međutim i dalje se veoma teško i neefikasno izvodi. Mnoga izvješća i istraživanja o iskustvima onih koji su poučavali objektno orijentirano programiranje uključuju dugu listu problema s kojima su se susreli i to s mnogo različitih aspekata. Zašto je objektno orijentirano programiranje tako teško za poučavanje? Ili preciznije, zašto se poučavanje objektno orijentiranog programiranja čini teže nego poučavanje proceduralnog programiranja? Prije nego što se odgovori na to pitanje, treba sagledati drugi aspekt ovog problema, a on glasi „Kada se treba započeti sa učenjem objektno-orijentiranog programiranja?“

Dugo vremena se smatralo da je objektno-orijentirano programiranje napredni subjekt programiranja koji se uči kasno u kurikulumu. To se danas polako mijenja: sve više i više sveučilišta počinju poučavati objektno orijentirano usmjerenje i koncepte u svom prvom kolegiju za programiranje. Glavni razlog za to je često citiran problem pomaka paradigmi. Učenje objektno orijentiranog stila programiranja čini se vrlo teško nakon što se navikne na korištenje proceduralnog stila programiranja. Istraživanja pokazuju da je prosječnom programeru potrebno od 6 do 18 mjeseci za prebacivanje stanja uma iz proceduralnog u objektno orijentirani pogled na svijet. Iskustva, s druge strane, također pokazuju da učenicima nije tako teško razumjeti objektno orijentirana načela, ukoliko se s njima susretnu odmah na početku. Zapravo, prebacivanje s jednog stila na drugi je ono što je teško, a ne objektno-orijentirani stil programiranja. Za poučavanje programiranja, pouka je jasna: ako se želimo poučavati objektno orijentirano programiranje, to se treba učiniti u prvom programskom kolegiju. Put do objektno-orijentiranog stila programiranja kroz proceduralno programiranje je zapravo nepotrebno kompliciranje. Učenici tada prvo moraju naučiti jedan stil programiranja, zatim ga moraju zaboraviti ili najblaže rečeno modificirati prethodno naučeno, prije nego što im pokaže kako to učiniti na pravi način.

Također, mnogo ljudi gleda na objektno-orijentirano programiranje kao na učenje još jednog programskog jezik (npr. prijelaz s C programskog jezika na C++ i slično) u kojem se mogu izvoditi novi koncepti nakon struktura, pokazivača i rekurzije. To je velika pogreška. Prema mišljenju većine, objektno-orijentirano programiranje ne uzrokuje

probleme nego dostupni alati za poučavanje istog. Programski jezici su previše složeni dok su programska okruženja za rad u određenom jeziku previše zbunjujuća za početnika. Zaključno, razlog za sve nevolje su pogrešni jezici i okruženja koja se koriste. To naravno dovodi do sljedećeg pitanja a ono glasi „Koje alate koristiti?“ Moje osobno mišljenje je da je trenutno u svijetu najbolji jezik za početnike i općenito za poučavanje objektno orijentiranog programiranja svakako Python. C# programski jezik svakako nije preporučljiv za prvi kontakt s programiranjem, ali kako se danas jako često koristi pri samom je vrhu pogodnih jezika za objektno orijentirano programiranje, odmah poslije Pythona, C++ i Jave. Da bi određeni programski jezik bio pogodan za poučavanje objektno orijentiranog programiranja treba ispunjavati sljedeće kriterije:

- Jasni koncepti,
- Čisto objektno orijentirano usmjerenje,
- Sigurnost,
- Jezik visoke razine,
- Jednostavan „objekt/izvršavanje“ model,
- Lako čitljiva sintaksa,
- Neredundancija,
- Kompaktan programski jezik,
- Jednostavan prijelaz na druge programske jezike,
- Podrška za ispravno sigurnost koda,
- Ugodno radno okruženje

Istraživanja oko programskih jezika najčešće korištenih za poučavanje objektno orijentiranog programiranja u nastavi pokazuju da su u praksi prisutni problemi sa svakim od programskih jezika. Problemi su drugačije prirode za svaki od jezika. Iz svih istraživanja može se izvući zaključak da je objektno orijentirano programiranje viđeno kao nedvojbeno snažan i značajan alat za početnike, dok su skoro svi nastavnici u istraživanjima prijavili poteškoće sa određenim programskim jezicima i sustavima koje su koristili. Većina istraživanja je prijavila i poteškoće kod prijelaza s proceduralnog na objektno orijentirani način razmišljanja kod učenika.

U analizi programskih jezika, došlo se do zaključka da programski jezik zajedno sa programskim okruženjem mora biti uzet u obzir pri odabiru jezika za poučavanje objektno-orijentiranog programiranja. Dobro dizajniran programski jezik je samo polovica od onog

potrebnog i može se smatrati beskorisnim u odsustvu odgovarajućeg radnog okruženja. Kod nekih programskih jezika, radno okruženje je predstavljalo izvor većine ozbiljnih problema. Tako da se može zaključiti da ne postoji idealan programski jezik za poučavanje objektno-orijentiranog programiranja.

Ideja ovog rada je implementirati u GIFT C# aplikaciju za poučavanje objektno orijentiranog programiranja u programskom jeziku C#. C# aplikacija sadrži osnovne lekcije te u sebi ima integriranu provjeru razumijevanja lekcija prije nego dozvoli učeniku da rješava zadatke. Glavni cilj C# aplikacije je na što jednostavniji način objasniti učeniku određenu lekciju bez spominjanja nepotrebnih koncepata. Kroz C# aplikaciju se pokušalo iz vlastitog iskustva i poteškoća u startu kod učenja objektno orijentiranog programiranja uzeti u obzir one problematične faktore, razložiti ih, jednostavno objasniti te na taj način što više olakšati osobi koja tek pristupa tom načinu razmišljanja.

4.2. C# programski jezik

C# ima za prethodnike programske jezike C i C++ od kojih je preuzeo sve stvari koje su bile dobre i koje nisu zahtijevale poboljšanja. Izrazi, naredbe i skoro cijela sintaksa, što čini većinu tipičnih programa je ostala nepromijenjena. Zbog toga se C i C++ programeri kad počnu učiti C# osjećaju kao na „domaćem terenu“. Učinjen je veliki napredak i dodana su mnoga poboljšanja i nadogradnje koje ne bi imalo smisla sve nabrajati. Neke od nadogradnji koje su uklonile dosta česte i vremenski zahtjevne pogreške u C i C++ programima su sljedeće:

- Varijable se moraju inicijalizirati prije nego što se počnu koristiti - nema više grešaka da se može koristiti neinicijalizirana varijabla.
- Naredbe `if` i `while` zahtijevaju Boolean vrijednosti tako da programeri više ne mogu greškom koristiti operator „=“ umjesto operatora „==“.
- Naredba `switch` više ne podržava prolazak na sljedeći case blok bez eksplicitnog navođenja, što je prije stvaralo pogreške ako programer zaboravi staviti break naredbu na kraju bloka.

Od bitnijih novosti može se navesti sljedeće:

- C# ima integrirano svojstvo automatskog upravljanja memorijom (eng. Garbage collector) a njegova glavna zadaća je da olakšava programerima kod

oslobađanja memorije. "Viseći pokazivači" i neoslobađanje memorije su neke od najčešćih grešaka u C++ programskom jeziku koji se uvelike bazira na rad sa pointerima. Pri tome se nije izgubilo na funkcionalnosti jer se još uvijek može raditi sa pokazivačima i adresama objekata u rijetkim slučajevima kad se baš mora ručno upravljati memorijom.

- Svi podaci su prezentirani kroz objekte. Premoštena je razlika između value i reference tipova, tako da se svi podaci mogu obrađivati kao objekti.
- U programskom jeziku C# u klasama se mogu definirati svojstva (properties), koja omogućavaju kontrolirani i intuitivniji pristup podacima koji se nalaze u objektu nego preko getter i setter metoda.
- C# programski jezik podržava atribute koji omogućavaju definiranje i korištenje deklarativnih informacija o komponentama. Mogućnost definiranja nove deklarativne informacije je moćan alat koji otvara programerima nove, i prije teško izvedive, mogućnosti.

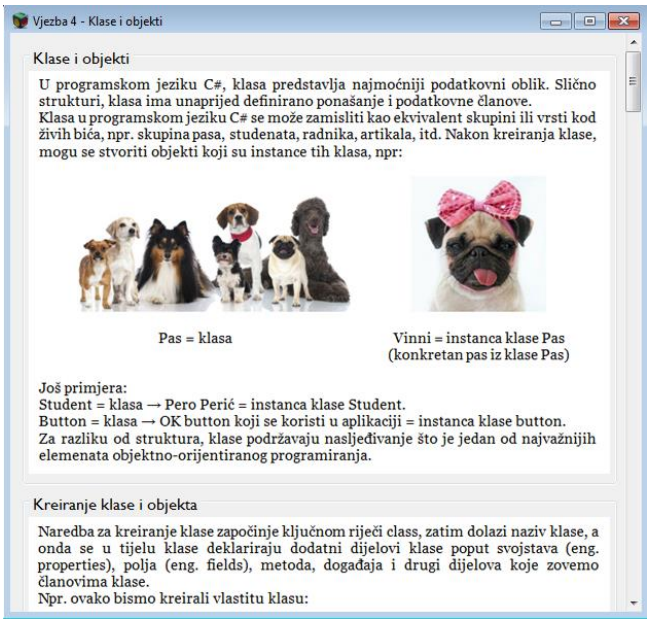
Kao takav C# je danas jedan od vodećih programskih jezika kad je u pitanju objektno orijentirani pristup. Odabran je iz razloga što se danas na fakultetima jako često koristi, kako u kolegijima tako i za izradu projektnih zadataka i pogodna je osnova za upoznavanje sa objektno orijentiranim pristupom.

5. Aplikacija za poučavanje objektno-orijentiranog programiranja

C# aplikacija je zamišljena na način da, osim što objasni osnovnu sintaksu, ponudi i odgovor na pitanje zašto se koristiti određena mogućnost. Ako do sada učenik nije koristio sučelja ili klase, puko navođenje sintakse ga sigurno neće nagovoriti da koristi mogućnosti koje mu oni nude. Zbog toga se u svakom poglavlju i za svaki primjer nalazi i kôd koji objašnjava koje probleme u programima ta mogućnost pokušava riješiti. U nekim lekcijama koje nije moguće tek tako jednostavno objasniti davanjem primjera, lekcija je sačinjena na način da vodi učenika uputu po uputu kako da napravi testnu aplikaciju na kojoj će naučiti taj novi koncept. Npr, jedna od tih lekcija su sigurno i „Iznimke“. Ova C# aplikacija je napravljena s namjerom da korisnika upozna s OOP konceptima kroz jednostavne primjere, pojednostavljene definicije i lagana pitanja koja su ključna za znati prije kodiranja. Zamišljeno je da korisnik prvo prouči materijale za određenu nastavnu jedinicu koristeći GIFT, a po završetku GIFT će ga preusmjeriti na C# aplikaciju koja ima zadaću da korisnik dodatno utvrdi svoje znanje kroz lagane vježbe sa pitanjima iza kojih slijedi tranzicija na izradu zadataka. Materijali u GIFT-u su napisani na engleskom jeziku, dok je C# aplikacija za poučavanje OOP-a na hrvatskom jeziku. To je napravljeno s namjerom jer engleski jezik se mora znati aktivno koristiti u programiranju. Kad god se pojavi neki problem ili greška u kodu, danas je nemoguće pronaći rješenje na internetu ukoliko se ne poznaje engleski jezik I terminologija koncepata za koji se traži rješenje. Hrvatski jezik je odabran za aplikaciju iz razloga što je tu sve objašnjeno do u detalje s namjerom da korisnik razumije svaku riječ I svaki od primjera koda, koji su također napisani na hrvatskom jeziku.

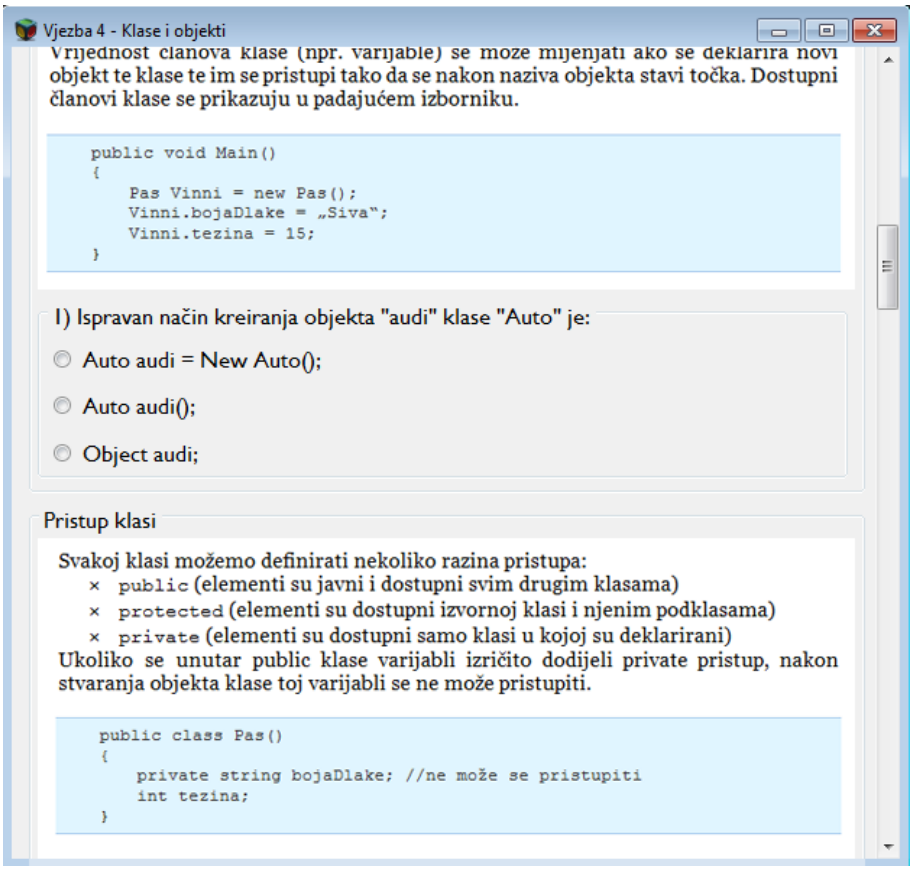
Slika 8 Glavna početna forma C# aplikacije

Aplikacija ima 2 glavne forme, to su prva početna forma koja sadrži sve nastavne jedinice i druga forma koja sadrži sve zadatke za pojedinu nastavnu jedinicu. Na klik dugmeta otvara se forma koja je „skrolabilna“ na kojoj je objašnjena odgovarajuća nastavna cjelina.



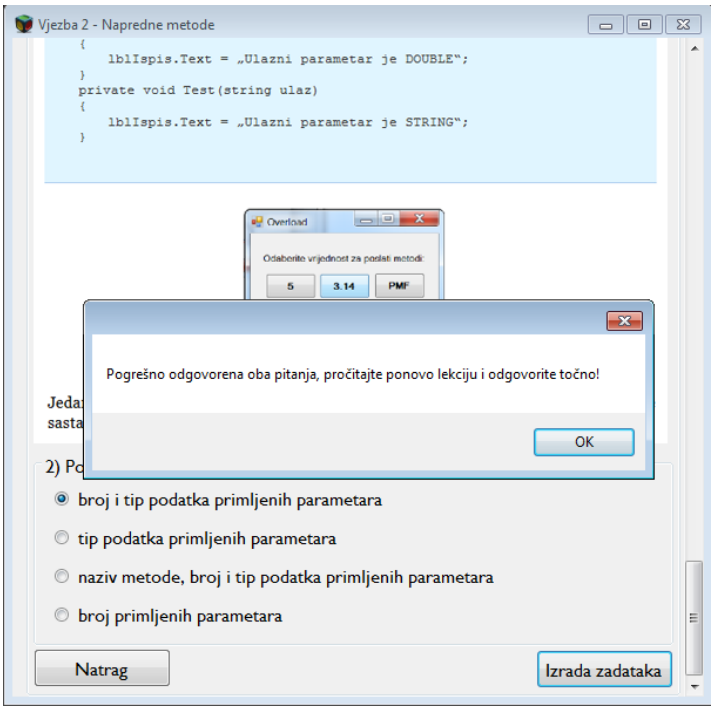
Slika 9 Forma vježbe "Klase i objekti"

Pitanja su ubačena na prijelomnim točkama svakog važnijeg koncepta u toj nastavnoj cjelini. Većinom su to 2 do 3 pitanja po nastavnoj jedinici, a broj pitanja nikada ne prelazi 4. Pitanja nisu teška, štoviše jako su jednostavna i mogu se odgovoriti ukoliko se tekst vježbe pročita s razumijevanjem, a koncipirana i odabrana po važnosti, jer korisnik mora znati odgovore na ta pitanja kako bi mogao pristupiti izradi zadataka i uspješno ih riješiti.



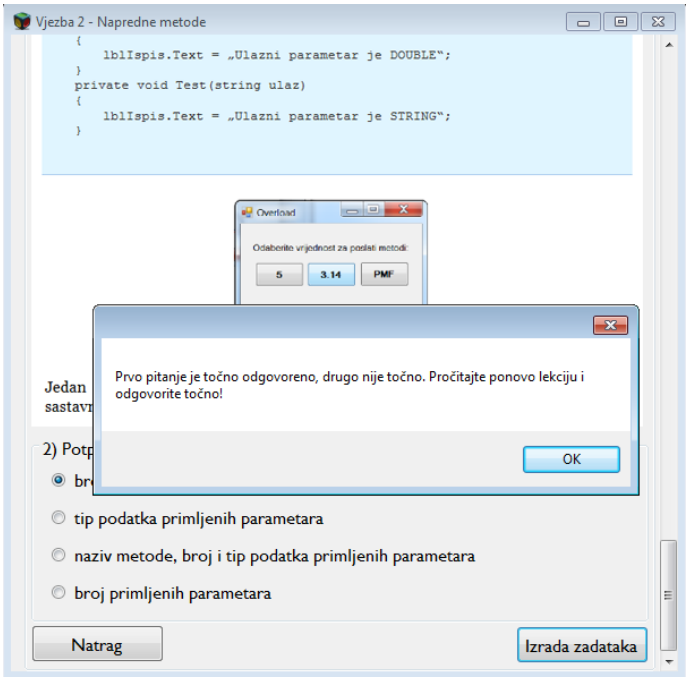
Slika 10 Primjer jednog od pitanja u vježbi

Kod izrade ovih pitanja, uzelo se u obzir to što su danas u većini kolegija učenici/studenti pretrpani sa informacijama koje i nisu baš od ključne važnosti, a koje je svakako dobro znati u nekim daljnjim fazama programiranja. Ipak, jednom početniku je važnije znati kako da kreira novi objekt neke klase nego da zna više vrsta načina na koji se to može napraviti. To na neki način dovodi i do pogrešnih poimanja u programiranju, tako da se u ovom radu to nastojalo izbjeći što je više moguće. Korisnik mora odgovoriti na sva pitanja točno kako bi mogao preći na izradu zadataka.



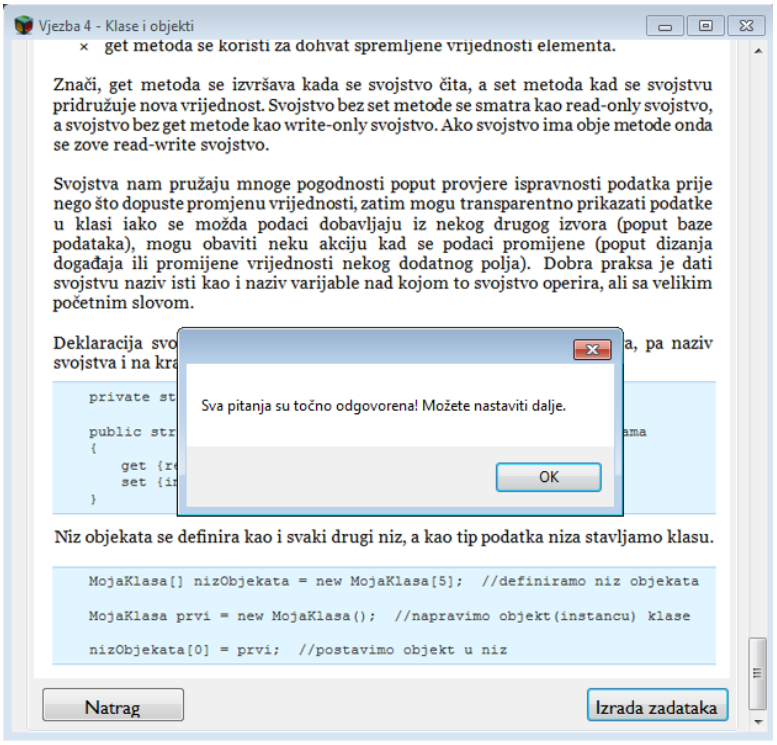
Slika 11 - Poruka kod pogrešnih odgovora

Ukoliko korisnik ne odgovori točno na sva pitanja ili samo na neko od pitanja sustav mu ne dopušta da ide dalje, već ga upućuje da pročita gradivo još jednom i točno odgovori na pitanja.



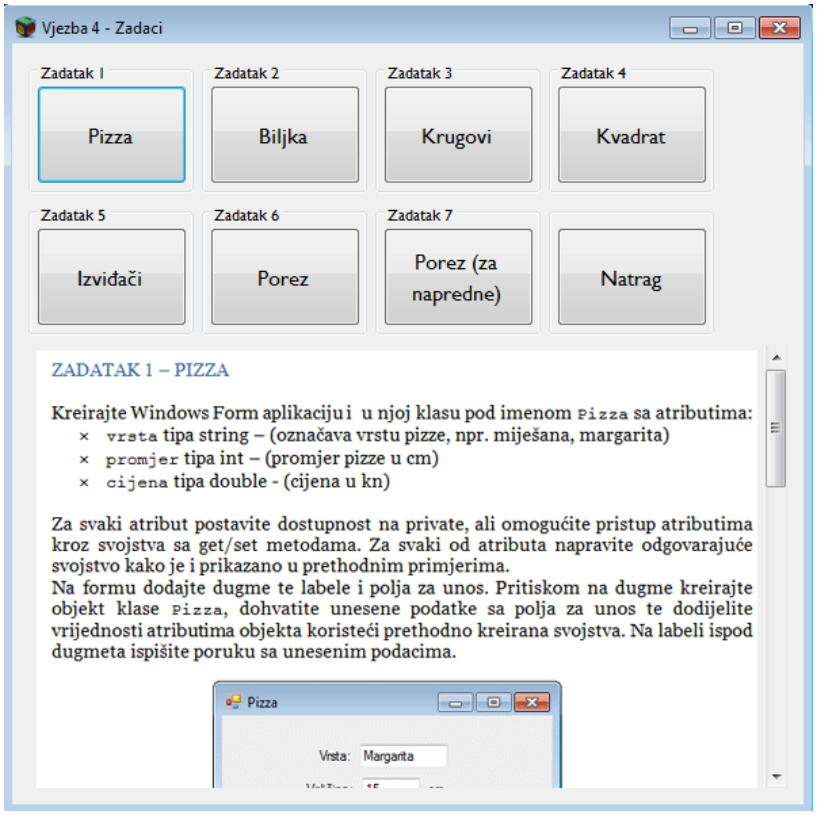
Slika 12 - Poruka kod djelomično točnih odgovora

Nakon što korisnik točno odgovori na sva pitanja, otvara mu se prozor sa zadacima i on ima slobodan izbor da rješava koji god zadatak želi, redoslijed nije bitan.



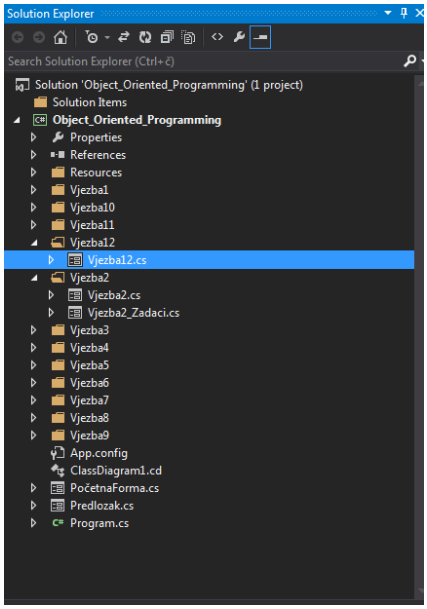
Slika 13 Poruka C# aplikacije ukoliko su svi odgovori točni

Korisniku se otvara prozor sa zadacima:



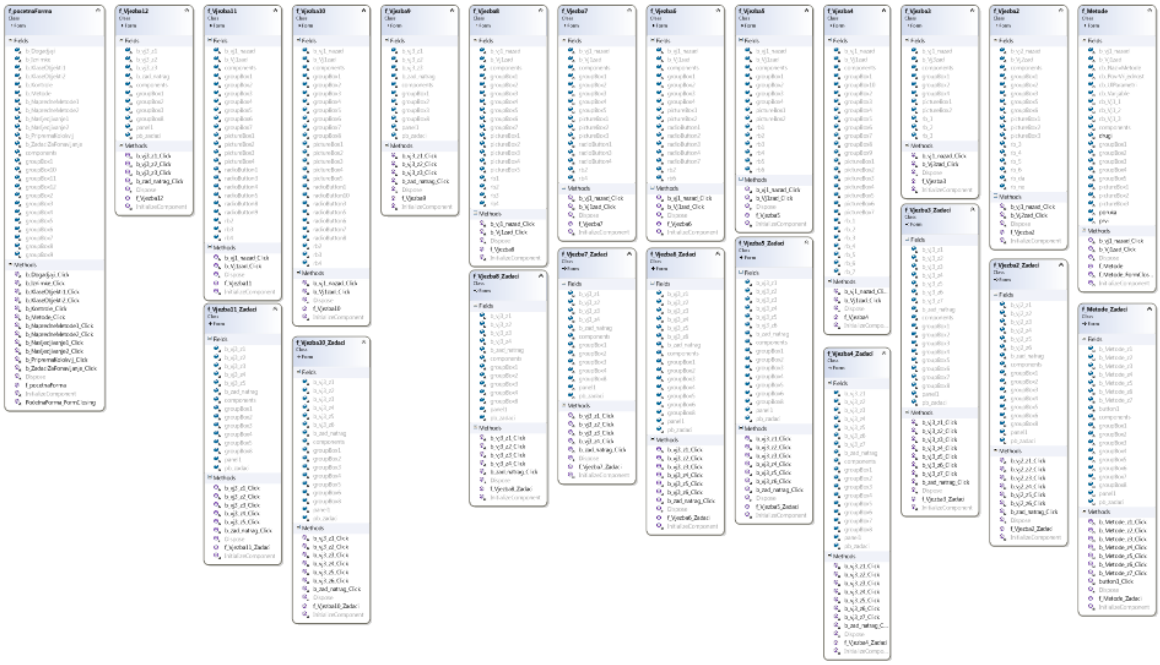
Slika 14 Forma sa zadacima

Zamišljeno je da nakon uspješno riješenih zadataka korisnik svom nastavniku šalje zadatke mailom, Dropbox-om ili ih stavlja u svoj folder na mrežnom disku kako bi ih nastavnik mogao kasnije ocijeniti. Sama struktura klasa unutar projekta izgleda kao na slici:



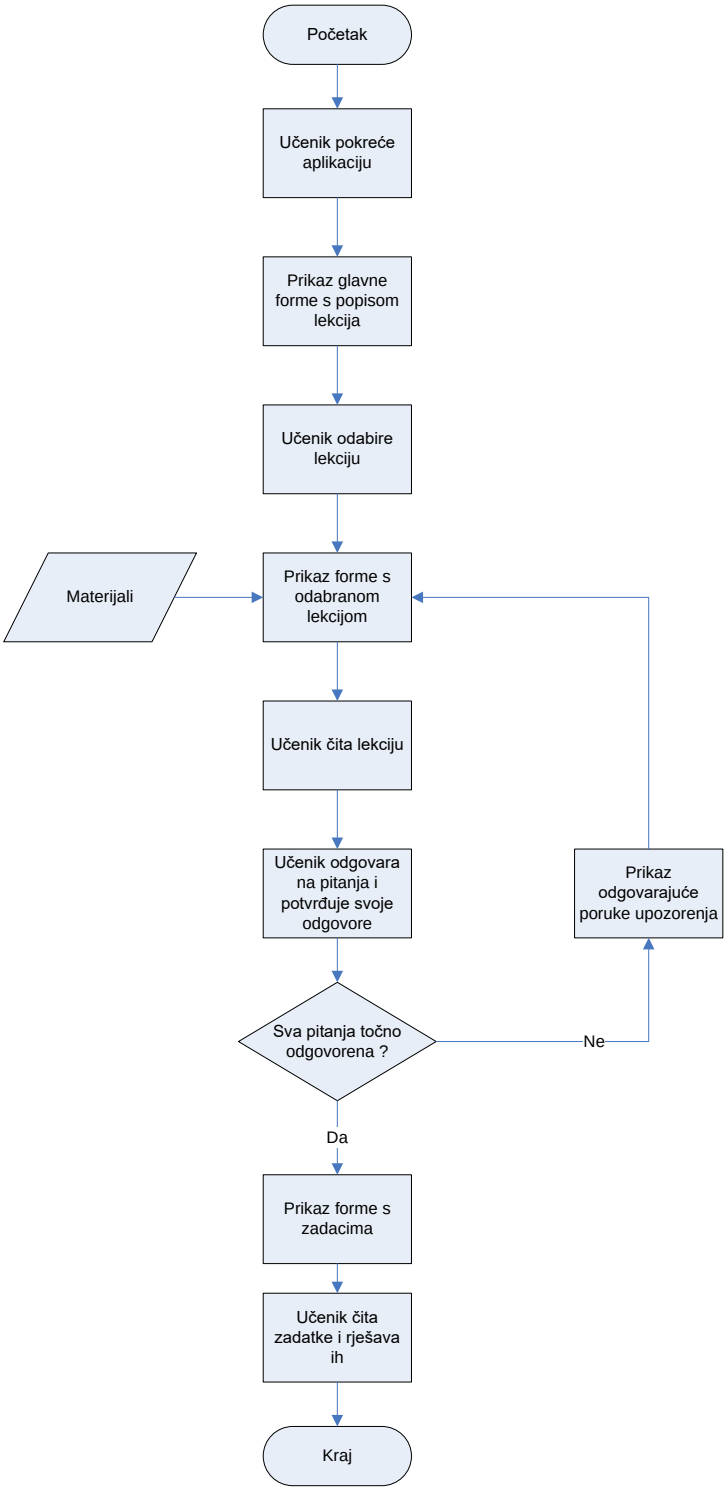
Slika 15 Projekt C# aplikacije

S druge strane sam dijagram klasa izgleda vrlo velik s obzirom na broj klasa i implementiranih elemenata u klasama:



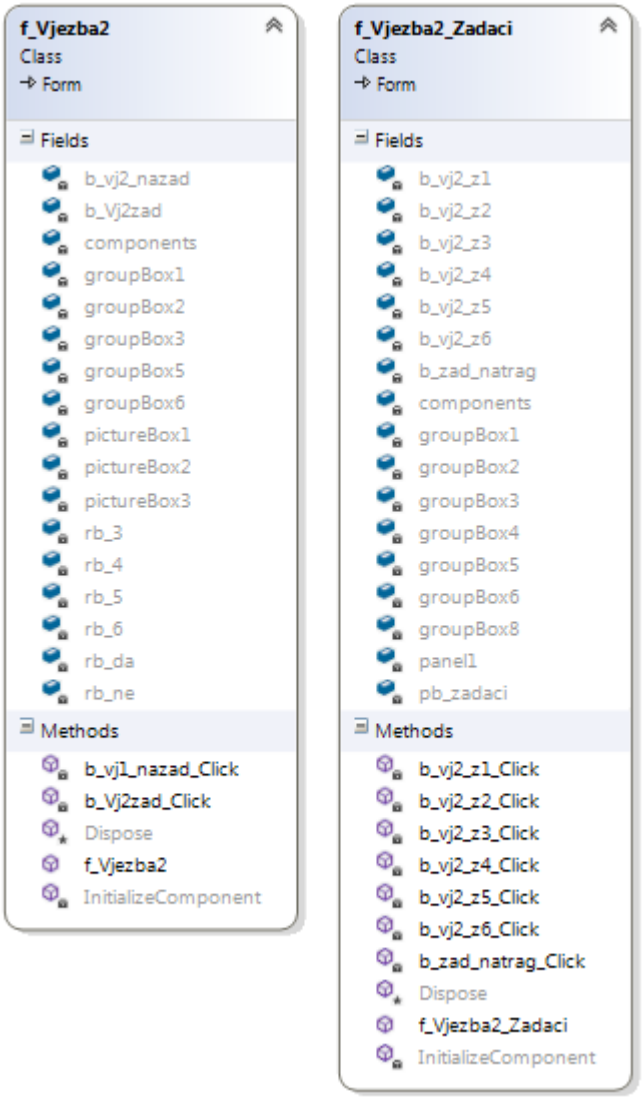
Slika 16 Dijagram klasa

Svaka vježba odnosno klasa je napravljena po modelu „vježba/zadaci“, pa su tako sve forme i strukturirane. Radni dijagram (eng. workflow diagram) aplikacije je prikazan na sljedećoj slici:



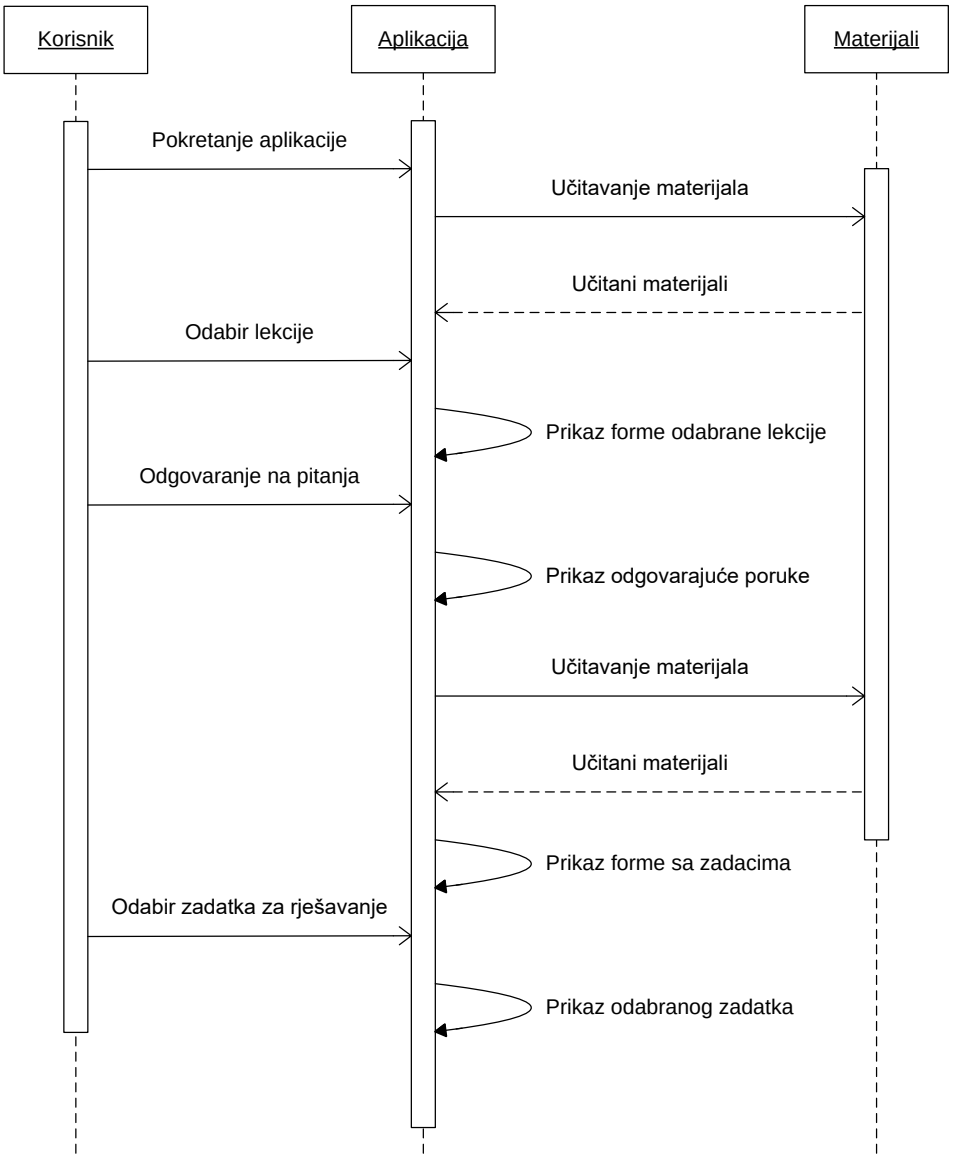
Slika 17 Radni (workflow) dijagram C# aplikacije

Malo jasniji uvid u strukturu klasa je prikazan na sljedećoj slici koja prikazuje primjer za Vježbu 2 – „Napredne metode“:



Slika 18 Dijagram klasa za vježbu 2 "Napredne metode"

Svaka aplikacija ima svoj slijed aktivnosti i sekvenci koje se mogu prikazati dijagramom slijeda:



Slika 19 - Dijagram slijeda C# aplikacije

.

Zaključak

GIFT kao alat je veoma mlad i pred njim je definitivno još puno godina „sazrijevanja“. Kako je danas softverska konkurencija po pitanju dostupnosti, raznovrsnosti, marketinške strane, optimalnosti i drugih faktora neumoljiva i daleko ispred GIFT-a, potrebno je sprovesti određene mjere kako bi GIFT postao alat koji odgovara današnjem prosječnom korisniku koji nema završen studij informatičke struke i koji ga može koristiti u svakodnevnim aktivnostima. Konkretno, GIFT je u ovom trenutku dostupan za korištenje svima i kao sustav ima određene prednosti koje su opisane u ovom radu a koje drugi sustavi nemaju, međutim u određenim segmentima uvelike zaostaje za današnjim vodećim alatima koji se koriste u svrhe poučavanja poput Moodle-a i drugih. Ti segmenti u kojima GIFT zaostaje su defitivno uzrokovani lošim počecima zbog toga što je GIFT prvotno bio namijenjen kao sustav za korištenje u vojne svrhe (obučavanje, ratni treninzi, predavanja) da bi nakon nekog vremena rukovodioci projekta odlučili da ga transformiraju u sustav koji je polivalentan odnosno dostupan za sve moguće oblike poučavanja u svim domenama ljudskih djelatnosti koji u sebi ima integriranu podršku za komunikaciju s drugim sustavima kao što su video igre, simulacije, virtualna realnost i fizički uređaji poput senzora za mjerenje određenih aktivnosti korisnika i obrade podataka.

Ono je najviše frustrirajuće u radu s GIFT-om je tromost sustava. Testirano na Acer laptopu s intelovim i5 procesorom, 4 GB RAM memorije i operativnim sustavom Windows 7 u 10 mjerenja prosječno trajanje osvježavanja glavne GIFT stranice s kolegijima iznosi 56.3 sekunde. Ovo vrijeme čekanja za učitavanje kolegija je proporcionalno količini napravljenih GIFT kolegija (više kolegija = duže vrijeme osvježavanja stranice). GIFT tijekom pokretanja troši oko 485000 Kb ~ 500 MB RAM memorije, što je skoro 2x više od Google Chrome web preglednika. Naravno riječ je o GIFT desktop aplikaciji. Pored desktop varijante od 03.11.2015 dostupna je i cloud verzija GIFT-a s identičnim funkcijama kao i desktop ali sa puno boljim i bržim performansama u odnosu na GIFT desktop verziju. Za desktop verziju bi također trebalo napraviti i bolji način pokretanja GIFT programa u odnosu na trenutni način gdje se GIFT pokreće preko .bat skripte koja se nalazi u instalacijskom direktoriju GIFT-a. Također i svi drugi alati i moduli se nalaze dobro skriveni unutar instalacijskog direktorija. Što se tiče

dokumentacije, dostupna je preko web stranice www.gifttutoring.org te je jako detaljna i konkretna po pitanju objašnjavanja određenih stvari. Po pitanju dokumentacije za programere i developere, trebalo bi dokumentaciju koja se odnosi na programiranje dodataka za pokretanje C# aplikacija i općenito programiranje novih plugin-a staviti u jedan zaseban dokument. Trenutno je raspršena na više odvojenih dokumenata pa je malo teže pronaći željeni dio. Pored toga bilo bi zgodno omogućiti korisnicima sortiranje kolegija na glavnoj stranici GIFT-a po njihovim kriterijima radi lakše organizacije. Također neke od funkcionalnosti GIFT-a nisu radile kako je predviđeno (npr. učitavanje C# aplikacije u određenom trenutku nakon poziva akcije za prikaz PDF dokumenta), ali su sada popravljene i uredno rade u novoj verziji GIFT-a koja je izašla 03.11.2015. godine.

Neke od gore opisanih sugeriranih promjena koje bi definitivno poboljšale kvalitetu GIFT-a i općenito zadovoljstvo korisnika su najavljene za sljedeću verziju GIFT-a koja bi uskoro trebala ugledati svjetlo dana.

Zaključno, može se reći da GIFT definitivno ima potencijal da bude jedan od vodećih sustava na području poučavanja. Ukoliko se uzmu u obzir okolnosti pod kojima je nastao, glavna namjena, zatim prenamjena GIFT sustava, dosadašnje postignute rezultate, glavne funkcionalnosti GIFT-a te da ARL osoblje uporno radi poboljšanju istih, za očekivati je da GIFT postane veoma popularan sustav u bližoj budućnosti. To prvenstveno vrijedi za korisnike koji rade u obrazovnim domenama (nastavnici, profesori, demonstratori, treneri itd.), a zatim i za sve ostale vrste korisnika.

Literatura

[1] Charles Ragusa, Michael Hoffman, and Jon Leonard - „Unwrapping GIFT, A Primer on Developing with the Generalized Intelligent Framework for Tutoring“

[2] Robert A. Sottilare, Anne M. Sinatra - „Proceedings of the 3rd Annual Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym3)“

[3] R. Sottilare, A. Graesser, X. Hu, and K. Brawner – „Design Recommendations for Intelligent Tutoring Systems: Volume 3 - Authoring Tools and Expert Modeling Techniques“, Orlando, FL: U.S. Army Research Laboratory.

[4] GIFT source documentation: Generalized Intelligent Framework for Tutoring Authoring Guide 4.1

[5] GIFT source documentation: Generalized Intelligent Framework for Tutoring Developers Guide 3.0

[6] Robert Sottilare Arthur Graesser Xiangen Hu Heather Holden - „Design Recommendations for Intelligent Tutoring Systems Volume 1 Learner Modeling“

[7] Gul Agha, Peter Wegner, Akinori Yonezawa - „Research Directions in Concurrent Object-Oriented Programming“, March 2003

[8] OLE LEHRMANN MADSEN – „Open Issues in Object-Oriented Programming, A Scandinavian Perspective - Computer Science Dept.“, Aarhus University, Ny Munkegade , DK-8000 Aarhus C, Denmark

[9] D.C. Tsichritzis, O.M. Nierstrasz – „Directions in Object-Oriented Research“, chapter 20

[10] Chris Hostetter – „Survey of Object Oriented Programming Languages“ - 1998-05-23

[11] William R. Cook - „Object-Oriented Programming Versus Abstract Data Types“, Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA, 94303-0969, USA

[12] Michael Kölling – „The problem of teaching object-oriented programming - Part I: Languages“ - School of Computer Science and Software Engineering, Monash University

[13] Jens Bennedsen, Michael E. Caspersen – „Teaching Object-Oriented Programming towards Teaching a Systematic Programming Process“, IT University West, Fuglesangs Allé 20, DK-8210 Aarhus V, Denmark &, IT University West, University of Aarhus, IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark

[14] Davis, F. D. - Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly, 13*(3), 319-340.

[15] Nielson, D. - Ten Usability Heuristics for User Interface Design. Retrieved April 2014

[16] Norman, D. - *The Design of Everyday Things*. New York, NY: Basic Books.

[17] Preece, J., Rogers, Y. & Sharp, H. - *Beyond Interaction Design: Beyond Human-Computer Interaction*: John Wiley & Sons, Inc.

[18] Shneiderman, B., Plaisant, C., Cohen, M. & Jacobs, S. - *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (5th ed.). USA: Prentice Hall.

[19] Lesgold, A.M., Lajoie, S., Bunzo, M. & Eggan – "A coached practice environment for an electronics trouble shooting job", LRDC Report. Pittsburgh, PA: University of Pittsburgh, Learning Research and Development Center.

[20] Murray, T. – "Authoring intelligent tutoring systems: An analysis of the state of the art", *International Journal of Artificial Intelligence in Education*, 10(1):98–129.

[21] Murray, T. – "An Overview of Intelligent Tutoring System Authoring Tools: Updated analysis of the state of the art", *Authoring tools for advanced technology learning environments*. 2003, 491-545.

[22] Sottilare, R. and Gilbert S. – "Considerations for tutoring, cognitive modeling, authoring and interaction design in serious games", Authoring Simulation and Game-based Intelligent Tutoring workshop at the Artificial Intelli-gence in Education Conference (AIED) 2011, Auckland, New Zealand, June 2011.

[23] Sottilare, R., Brawner, K., Goldberg, B. & Holden, H. – "The Generalized Intelligent Framework for Tutoring (GIFT)", US Army Research Laboratory.

[24] Sottilare, R., Goldberg, B., Brawner, K. & Holden, H. – "A modular framework to support the authoring and assessment of adaptive computer-based tutoring systems (CBTS)", In Proceedings of the *Interservice/Industry Training Simulation & Education Conference*, Orlando, Florida, December 2012.

[25] Sottilare, R., Holden, H., Goldberg, B. & Brawner, K. – "The Generalized Intelligent Framework for Tutoring (GIFT)", In Best, C., Galanis, G., Kerry, J. and Sottilare, R. (Eds.) *Fundamental Issues in Defence Simulation & Training*. Ashgate Publishing.

[26] VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L. – "The Andes physics tutoring system: Lessons learned", *International Journal of Artificial Intelligence and Education*, 15(3), 147–204.

[27] VanLehn, K. – "The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems", Educational Psychologist, 46:4, 197-221

[28] Vygotsky, L.S. – "Mind in Society: The development of higher psychology processes", Cambridge MA: Harvard University press.