USPOREDNA ANALIZA METODA RAZVOJA INFORMACIJSKIH SUSTAVA

Konopek, Zrinka

Sveučilište Josipa Jurja Strossmayera u Osijeku
Ekonomski fakultet u Osijeku
Sveučilišni prijediplomski studij Poslovna informatika

Zrinka Konopek

USPOREDNA ANALIZA METODA RAZVOJA INFORMACIJSKIH SUSTAVA

Završni rad

Osijek, 2023.

Sveučilište Josipa Jurja Strossmayera u Osijeku
Ekonomski fakultet u Osijeku
Sveučilišni prijediplomski studij Poslovna informatika

Zrinka Konopek

USPOREDNA ANALIZA METODA RAZVOJA INFORMACIJSKIH SUSTAVA

Završni rad

Kolegij: Oblikovanje i implementacija IS-a

JMBAG: 0010233795
e-mail: zkonopek@efos.hr

Mentor: doc. dr. sc. Dario Šebalj

Osijek, 2023.

Josip Juraj Strossmayer University of Osijek
Faculty of Economics and Business in Osijek
Undergraduate Study Business informatics

Zrinka Konopek

COMPARATIVE ANALYSIS OF INFORMATION SYSTEMS DEVELOPMENT METHODS

Final paper

Osijek, 2023.

IZJAVA

O AKADEMSKOJ ČESTITOSTI, PRAVU PRIJENOSA INTELEKTUALNOG VLASNIŠTVA, SUGLASNOSTI ZA OBJAVU U INSTITUCIJSKIM REPOZITORIJIMA I ISTOVJETNOSTI DIGITALNE I TISKANE VERZIJE RADA

1. Kojom izjavljujem i svojim potpisom potvrđujem da je završni rad isključivo rezultat osobnoga rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu. Potvrđujem poštivanje nepovredivosti autorstva te točno citiranje radova drugih autora i referiranje na njih.
2. Kojom izjavljujem da je Ekonomski fakultet u Osijeku, bez naknade u vremenski i teritorijalno neograničenom opsegu, nositelj svih prava intelektualnoga vlasništva u odnosu na navedeni rad pod licencom *Creative Commons Imenovanje – Nekomercijalno – Dijeli pod istim uvjetima 3.0 Hrvatska.* (cc) BY-NC-SA
3. Kojom izjavljujem da sam suglasan/suglasna da se trajno pohrani i objavi moj rad u institucijskom digitalnom repozitoriju Ekonomskoga fakulteta u Osijeku, repozitoriju Sveučilišta Josipa Jurja Strossmayera u Osijeku te javno dostupnom repozitoriju Nacionalne i sveučilišne knjižnice u Zagrebu (u skladu s odredbama Zakona o visokom obrazovanju i znanstvenoj djelatnosti, NN 119/2022).
4. izjavljujem da sam autor/autorica predanog rada i da je sadržaj predane elektroničke datoteke u potpunosti istovjetan sa dovršenom tiskanom verzijom rada predanom u svrhu obrane istog.

Ime i prezime studenta/studentice: Zrinka Konopek

JMBAG: 0010233795

OIB: 84780858744

e-mail za kontakt: zrinkaa081@gmail.com

Naziv studija: Poslovna informatika

Naslov rada: Usporedna analiza metoda razvoja informacijskih sustava

Mentor/mentorica rada: doc.dr.sc. Dario Šebalj

U Osijeku, 2023. godine

Potpis Konopek

Usporedna analiza metoda razvoja informacijskih sustava

SAŽETAK

Usporedbom različitih metoda razvoja informacijskih sustava identificiraju se njihove prednosti, nedostatci i prikladnosti pojedine metode za različite vrste IT projekata. Metode koje se analiziraju u ovom završnom radu su SDLC metoda, strukturna analiza, objektno-orijentirana analiza te agilne metodologije. Uspoređivat će se ključni aspekti svake metode, uključujući proces razvoja, fleksibilnost, vrijeme, troškove i kvaliteta konačnog informacijskog sustava. Istraživanjem se utvrđuje važnost odabira prikladne metode ovisno o specifičnostima projekta i okolnostima. Nadalje, istraživanje će analizirati ulogu timskog rada i suradnje u svakoj metodi. Efikasna komunikacija i dobra suradnja ključni su faktori uspjeha u razvoju informacijskih sustava. Također će se istražiti mogućnosti prilagodbe svake metode zahtjevima i promjenama tijekom razvojnog procesa, što može biti ključno za uspješnu isporuku informacijskog sustava koji zadovoljava potrebe klijenta. Uzimajući u obzir karakteristike svake metode, razvojni tim može odabrati najprikladniju metodu za postizanje uspješnog razvoja informacijskog sustava. Tijekom istraživanja, analizirani su i čimbenici koji utječu na uspješnost svake metode, uključujući timsku strukturu, komunikaciju i suradnju s klijentom te mogućnost prilagodbe zahtjevima i promjenama. Nakon razrade svake metode napravit će se usporedna analiza odabranih metoda po određenim kriterijima. Ovaj završni rad pruža dublje razumijevanje različitih metoda razvoja informacijskih sustava i pruža smjernice za odabir najprikladnije metode u konkretnim projektima.

Ključne riječi: informacijski sustav, informacije, analiza, razvoj

Comparative Analysis of Information Systems Developement Methods

ABSTRACT

By comparing different methods for developing information systems, it is possible to determine the advantages, shortcomings, and suitability of a particular method for different types of IT projects. The methods analysed in this paper are the SDLC method, structural analysis, object-oriented analysis, and agile methodologies. Key aspects of each method are compared, including the development process, flexibility, time, cost, and quality of the final information system. The research shows the importance of choosing an appropriate method depending on the specifics of the project and the circumstances. It also analyses the role of teamwork and cooperation in each method. Efficient communication and good cooperation are key factors for successful information system development. The ability to adapt to challenges and changes during the development process of each method is also examined, which can be critical to the successful delivery of an information system that meets a customer's needs. Considering the characteristics of each method, the development team can select the most appropriate method for successful information system development. The factors that influence the success of each method, such as team structure, communication and collaboration with the customer, and the ability to adapt to requirements and changes, were also analysed during the research. After explaining each method, a comparative analysis of the selected methods is performed according to specific criteria. This paper provides a deeper understanding of the different methods used to develop information systems and offers guidelines for selecting the most appropriate methods in each specific project.

Keywords: information systems, informations, analysis, development

Sadržaj

1. Uvod

Razvoj informacijskih sustava zahtijeva niz složenih koraka i procesa kako bi se osiguralo uspješno stvaranje, implementaciju i održavanje sustava. Kroz te faze važno je pratiti zahtjeve korisnika, osigurati kvalitetu i osigurati da sustav zadovoljava potrebe i ciljeve organizacije. Prilikom razvoja informacijskih sustava postoje različite metode razvoja.

Usporedna analiza metoda razvoja informacijskih sustava predstavlja važan koncept u području razvoja i implementacije informacijskih sustava. Oni su ključan dio organizacije jer omogućuju prikupljanje, pohranu i obradu informacija, što rezultira učinkovitosti i konkurentnosti organizacija. Ona pruža dublje razumijevanje različitih pristupa i tehnika koje se koriste u procesu razvoja i implementacije informacijskih sustava. Omogućuje organizacijama procjenu prednosti i nedostataka kako bi odabrali najprikladniju metodu za svoje specifične potrebe i ciljeve.

Usporedba metoda obuhvaća različite aspekte kao što su fleksibilnost, brzinu razvoja, prilagodljivost promjenama, troškove, kvalitetu i ostale aspekte. Pomoću toga identificiraju se najbolje prakse koje donose optimalne rezultate u razvoju informacijskih sustava.

Metode koje će se analizirati i na kraju usporediti su SDLC metoda, agilne metodologije, strukturne metode i objektno-orijentirane metode. U nastavku rada, pružit će se dublji uvid u svaku od navedenih metoda, analizirajući njihove prednosti i nedostatke, alate rada, načine rada i kroz primjere pobliže objasniti svrhu pojedine metode.

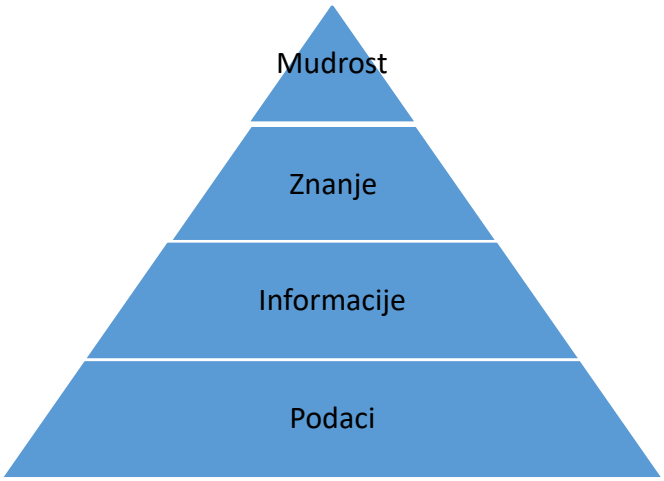
2. Informacijski sustavi

Danas su informacijski sustavi potrebni u svakom poslovnom subjektu kako bi se omogućilo kvalitetno i pravovremeno praćenje svih poslovnih procesa i olakšalo donošenje poslovnih odluka. Informacijski sustavi su temeljeni na informacijsko-komunikacijskoj tehnologiji, koja je postala neizbježan faktor današnjice. Temeljne funkcije informacijskih sustava prema Pavliću (2011) su prikupljanje i upis podataka u bazu podataka, obrada podataka, prikaz i ispostavljanje podataka iz baze podataka i čuvanje podataka. Prema istom autoru, informacijski sustav je skup povezanih dijelova (softver, hardver, ljudi, procedure, informacije te komunikacijske mreže) kojima je cilj pribaviti i prenijeti informacije i podatke za funkcioniranje, planiranje, odlučivanje i/ili upravljanje poslovnom organizacijom. Garača (2008) navodi da je hardver materijalna komponenta informacijskog sustava koje se sastoji od više grupa uređaja (centralne jedinice, ulazni i izlazni uređaji, vanjska memorija i drugo). Nadalje, softver definira kao nematerijalnu komponentu računalnih sustava koja sadrži znanje i iskustvo ljudi o načinima rješavanja problema iskazano u nizovima instrukcija i naredbi koje hardver može prepoznati i izvršiti. Softver se može podijeliti na dvije glavne skupine, a to su sistemski softver i korisnički (aplikacijski) softver. Sistemski softver je namijenjen olakšavanju korištenja računala a korisnički je namijenjen rješavanju konkretnih problema korisnika.

Sustav se promatra kao kompleksna mreža komunikacije između pojedinaca i skupina. Nije fokusiran samo na hijerarhijsku strukturu ili funkcionalne jedinice, već naglašava važnost razmjene informacija unutar neke organizacije. U takvom sustavu komunikacija igra ključnu ulogu u uspostavljanju veza i olakšavanju suradnje u internim poslovnim procesima. Kako bi informacijski sustavi mogli pravodobno raditi, treba prikupiti i obraditi veliki broj informacija. To se odvija pomoću računala, koje služi kao alat za obradu i prezentaciju informacija. Neki od procesa koji se provode u tom slučaju su primanje, upisivanje, obrada, distribucija, kontrola i druge operacije koje su moguće prilikom rada s informacijama (Pavlić, 2011).

Kao što je već spomenuto, informacijski sustavi sadrže neke ključne stavke, između ostalog i informacije, no ključni su i podaci, znanje i mudrost. Odnos podataka, informacija, znanja i mudrosti najbolje prikazuje piramidalna shema pod nazivom „DIKW“ (engl. *data, information, knowledge, wisdom*) piramida. Pavlić (2011) tvrdi da je podatak diskretna verzija neke fizičke veličine, skup znakova te da su podaci nositelji informacija. Bez podataka se ne može stvoriti vjerodostojna informacija. „Informacije dolaze iz odabranih podataka, tumačenja i predstavljanja u obliku koji za konkretnog primatelja u konkretnom kontekstu ima značenje i korisnost. Podatak postaje informacija kada je relevantan i razumljiv nekoj skupini ili

pojedincu“ (Pavlić, 2011:23). Nakon prikupljenih informacija koje su relevantne i razumljive stvara se znanje. Pavlić (2011) također definira znanje kao skup informacija s naputcima za djelovanje. Još navodi da je znanje informacija koja je testirana, potvrđena i kodirana. Samo po sebi znanje nosi vrijednost, tako da prilikom dijeljenja ono se ne gubi. Jedina mana dijeljenja znanja može biti slabljenje vrijednosti zbog proširenosti na tržištu koje smanjuje konkurentnost, navodi Palić. Na kraju piramide DIKW nalazi se mudrost, a ono predstavlja razumijevanje o tome koje se znanje upotrebljava s kojom namjenom.



Slika 1. "DIKW" hijerarhija

Izvor: Izrada autorice prema: Pavlić (2011)

3. Razvoj informacijskih sustava

Razvoj informacijskih sustava od ključne je važnosti za moderne organizacije kako bi se osigurala njihova uspješnost i konkurentska prednost. To je opsežan posao koji donosi mnoge prepreke i probleme, a neki od tih problema prema Garači (2008) su: nejasne potrebe poslovnog sustava, dinamične promjene poslovnih procesa, kontradiktorni zahtjevi krajnjih korisnika, nedostatna podrška menadžmenta, brzi razvoj informacijske tehnologije, neznanje korisnika što zaista žele od informacijskog sustava, itd. Razvoj informacijskih sustava je dugotrajan proces koji sadrži različite faze koje se mogu promatrati kao životni ciklus razvoja informacijskih sustava (Garača, 2008).

U daljnjim poglavljima opširnije će se proći kroz metode razvoja informacijskih sustava, od kojih će biti spomenute životni ciklus razvoja informacijskog sustava (engl. *Software Development Life Cycle* - SDLC), strukturna analiza, agilne metodologije, objektno-orijentiran analize, i druge značajnije metode razvoja.

3.1. Životni ciklus razvoja softvera (SDLC)

Metodologija životnog ciklusa razvoja softvera je najstarija metodologija, što ju čini temeljem razvoja drugih metodologija. Prema Pavliću (2009), poznata je i pod nazivom tradicionalna metodologija.

Za prikazivanje faza životnog ciklusa koriste se različite razine detaljnosti. Prema Garači (2008), osnovna podjela životnog ciklusa uključuje tri faze. To su analiza, dizajn i implementacija. Nazivi faza i njihov broj razlikuje se među autorima. Pa tako, primjerice, Pavlić (2011) navodi sljedeće faze:

1. Prethodno istraživanje
2. Analiza sustava
3. Dizajn sustava
4. Razvoj sustava
5. Implementacija sustava
6. Održavanje sustava

Faza analize informacijskog sustava obuhvaća utvrđivanje i specificiranje zahtjeva koji se postavljaju kao očekivanja od sustava koji se izgrađuje. U ovoj fazi radi se na modeliranju poslovnih procesa, modeliranju podataka i integraciji međusobnih veza. Detaljno se istražuju

činjenice. Neke od metoda analize su intervju, izravno promatranje, uzorci, proučavanje zapisa, upitnici i druge metode (Garača, 2008).

Dizajn obuhvaća oblikovanje informacijskog sustava, što obuhvaća dizajn njegove arhitekture, korisničkog sučelja, strukture baze podataka, i drugih. Prema Garači (2008), postoje dvije faze dizajna:

- Dizajn arhitekture, koji opisuje sustav na razini podsustava i aplikacijskih modula
- Detaljni dizajn, koji opisuje pojedine aplikacijske module i dizajn baze podataka

Prema Pavliću (2011:129), implementacija obuhvaćanje uvođenje novog hardvera i novog softvera uz kontrolu kvalitete. U ovoj fazi svi procesi odvijaju se postupno sve dok sustav nije u potpunosti zadovoljio zahtjeve.

Pavlić (2011) navodi da životni ciklus ima sljedeće dobre karakteristike:

- Koristi jednostavne metode
- Omogućio je komunikaciju analitičara i korisnika
- Dobro je ispitan i testiran
- Dobro je dokumentiran
- Osigurano je obrazovanje za njegove korisnike
- Osigurano je postizanje planiranih rokova
- Osigurano je držanje konačne cijene u granicama podnošljivosti
- Osigurano je dobro prikazivanje i ispunjenje korisničkih zahtjeva
- Moguće je praćenje napretka na projektu sa svih strana
- Omogućen je fazni razvoj
- Osigurano je korištenje alata i metoda u projektima

Također, Pavlić (2011) navodi i nedostatke SDLC-a:

- Opterećenost održavanjima
- Nezadovoljstvo korisnika
- Neuspjeh u zadovoljavanju potreba menadžmenta
- Neambiciozno oblikovanje novog sustava
- Procesi su nestabilni elementi u sustavu
- Konačni softverski proizvod je teško promjenjiv
- Problemi s dokumentacijom

- Nedostatak točnih procjena resursa
- Spor razvoj novih aplikacija
- Problemi s „idealnim“ pristupom

3.2. Strukturne metode

„Strukturne metode tretiraju funkcije i podatke manje više odvojeno, na način da se funkcije smatraju aktivnim komponentama sa svojstvom ponašanja, a podaci su pasivni elementi koji sadrže informacije i koji su predmet djelovanja funkcija“ Garača (2008). Ovakav pristup rezultira podijeljenosti sustava na funkcije u kojima se podaci međusobno razmjenjuju. Prilikom implementacije informacijskog sustava, funkcionalnosti se ostvaruju kroz programski kod, u kojem se detaljno opisuje način na koji se podaci pohranjuju.

Karakteristike koje ova metoda posjeduje su (Tutorialspoint, n.d.):

- Grafička metodologija, određuje prikaz aplikacije
- Procese dijeli tako da pruža jasan prikaz toka sustava
- Logička je, ne fizička, što znači da elementi sustava ne ovise o dobavljaču ili hardveru
- To je pristup koji funkcionira od općeg pregleda do detaljnih pojedinosti

Problem u ovoj metodologiji je kada tijekom održavanja sustava dođe do promjene sheme i formata podataka, jer to zahtijeva mijenjanje svih programa koji koriste te podatke. Strukturna metoda se uglavnom bazira na statičkoj strukturi sustava, što stvara poteškoće u prikazivanju dinamičkog ponašanja. Još jedan od nedostataka je zanemarivanje korisničkog sučelja, strukturna metoda je većinski fokusirana na analizu i dizajn sustava s tehničke perspektive (Ablison, n.d.).

Techopedia (2016) navodi glavne korake koji su prisutni u strukturnoj analizi:

- Proučavanje trenutne poslovne okoline
- Modeliranje starog logičkog sustava
- Modeliranje novog logičkog sustava
- Modeliranje novog fizičkog okruženja
- Evaluacija alternativa
- Odabir najboljeg dizajna
- Kreiranje strukturiranih specifikacija

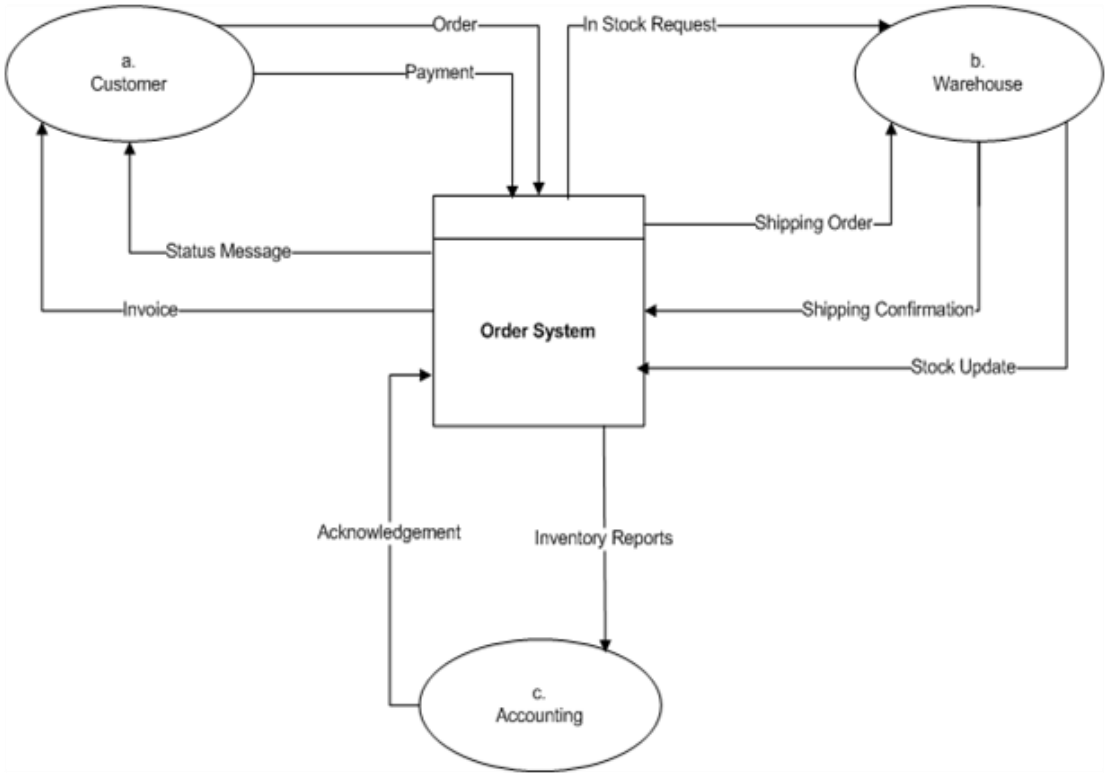
Neke od prednosti korištenja ove metode u razvoju informacijskih sustava su jasan prikaz podataka i procesa, što olakšava razumijevanje i komunikaciju članova i dionika. Strukturna analiza omogućava jasnu podjelu odgovornosti i zadataka unutar različitih komponenti sustava, što olakšava timski rad i omogućava efikasnije upravljanje projektom. Potiče se ponovna uporaba komponenti modula unutar sustava. Strukturna metoda sadrži još mnogo prednosti i nedostataka, bitno je napomenuti da su oni uvjetovani kontekstom i specifičnostima projekta (Ablison, n.d.).

Prema Tutorialspointu (n.d.), strukturna metoda ima različite alate kojima razvija sustave. Neki od tih alata su dijagrami protoka podataka, rječnik podataka, stablo odluka, tablice odluka, strukturirani engleski jezik, pseudokod i drugi alati. Neki od najkorištenijih dijagrama koji se koriste u ovoj metodi su dijagram toka podataka (DFD) i dijagrami struktura (SSD).

3.2.1. Dijagrami toka podataka (DFD)

Dijagrame toka podataka je razvio Larry Constantine kako bi grafičkim prikazom lakše prikazao potrebe sustava. Prikazuje se tok podataka između različitih funkcija sustava te način kako je trenutni sustav implementiran. To je inicijalna faza dizajna koja funkcionalno dijeli specifikacije zahtjeva do najnižih razina detalja. Sastoji se od simbola pomoću kojih se prikazuju iteracije događaja u procesu. Razlikuju se dvije vrste DFD dijagrama, to su logički dijagrami i fizički dijagrami (Tutorialspoint, n.d.).

Na slici 2 prikazan je DFD dijagram koja opisuje tok podataka između kupca, skladišta i računovodstva, prilikom naručivanja nekog proizvoda.

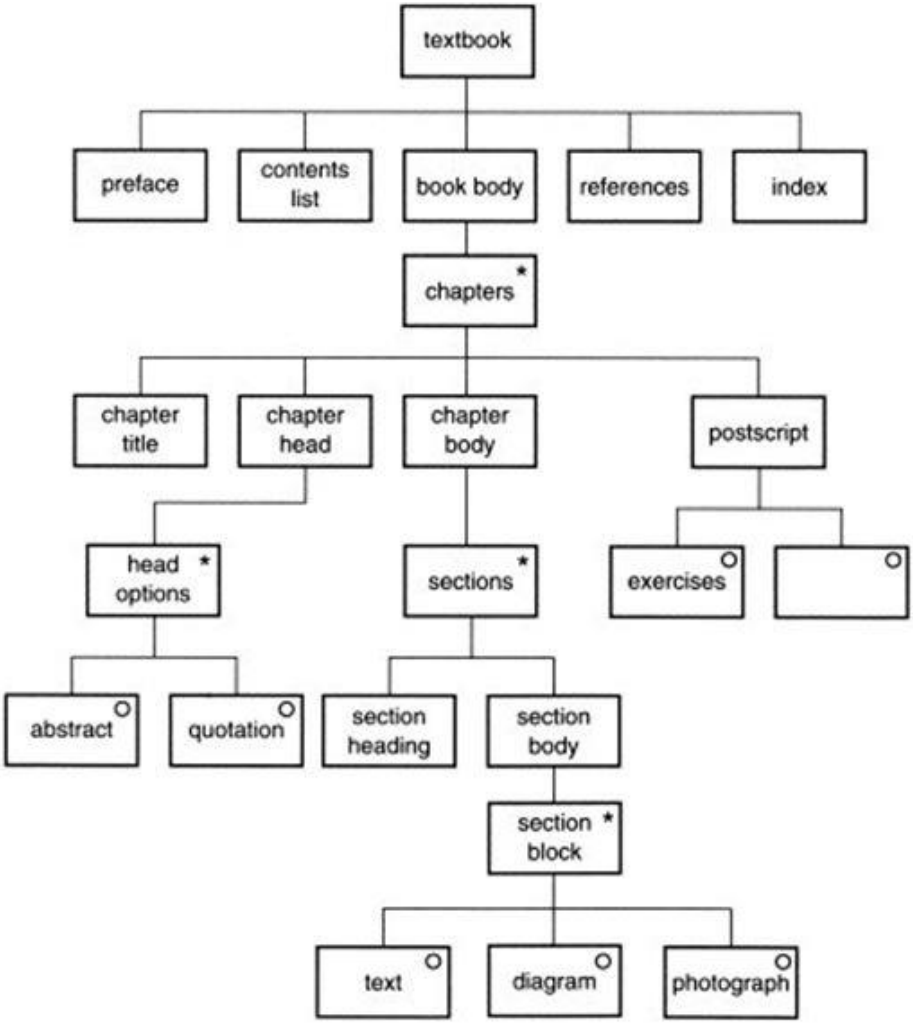


Slika 2. Primjer dijagrama toka podataka

Izvor: Eternal sunshine of the IS mind (n.d.)

3.2.2. Dijagrami struktura

Dijagrami struktura se koriste kada se želi prikazati interakcija između vanjskih korisnika i sustava. Opisuje korake koje korisnik izvodi kako bi ostvario određeni cilj. Koristi jednostavnu notaciju kojom se opisuje redoslijed događanja korisnika i sustava. Identificira i prikazuje sve moguće scenarije interakcija koje su moguće između ta dva aktera. Analitičarima omogućuje bolje razumijevanje toka interakcije i identificiranje ključnih koraka i poruka koje se razmjenjuju (Tutorialspoint, n.d.).



Slika 3. Dijagram struktura

Izvor: Link-eLearning (n.d.)

3.3. Objektno-orijentirana metoda

Pavlić (2011) govori o tome da objektno-orijentirana metodologija uvodi koncept objekta koji ujedinjuje podatke i operacije nad tim podacima te povezuje tradicionalne procesne modele i novije modele podataka u jedinstven model. Objektno-orijentirana analiza (u daljnjem tekstu OOA) opisuje informacijske sustave pomoću objekata kojima su dodane ljudske komponente, interakcije, događaji, zadaci i drugo. Output OOA je model objekta.

Objekt u terminologiji OOA predstavlja osobe, događaje, pojave… „Objekt je prikaz jedne zasebne jedinice koja može primiti ulaz, izvršiti proces i generalizirati izlaz.“ (Pavlić, 2011:281). Ako se spoji više objekata iste vrste govori se o klasi. Avison i Fitzgerald (1995) objašnjavaju odnos klase i objekta preko teorije Coada i Yourdona koji su 1991. dali tezu kako

je cijeli koncept objektno-orijentirane analize baziran na stvarima koje smo naučili kao djeca, objekti i atributi, cjeline i dijelovi, klase i članovi. Učimo identificiranjem cijelih objekata, kao na primjer drveće, a tek onda definiramo njihove atribute, kao što su lišće i grane. Nakon toga razlikujemo klase od objekta, kao što su drveće i kamenje. Ovaj pristup ukazuje na jednostavnost cijelog načina djelovanja OOA. Programski jezik kojeg je koristila ova analiza bio je Smalltalk, vrlo jednostavan programski jezik koji je originalno bio razvijen tako da ga mogu koristiti i djeca. Danas OOA integrira s programskim jezicima kao što su JavaScript, C++, Perl, Pyhton i drugi.

Garača (2008) spominje par aktivnosti kojima OOA objedinjeno izučava ponašanje sustava i informacija o sustavu:

- Pronalaženje objekata
- Grupiranje objekata
- Opisivanje veza
- Opisivanje ponašanja
- Opisivanje unutrašnjih svojstava

Ove aktivnosti su ključni koraci u procesu razvoja objektno-orijentiranih modela u problemskim domenama. Pažljivo izvršavanje tih koraka omogućuje precizno i strukturirano oblikovanje objekata kako bi se postigli željeni ciljevi u dizajnu i implementaciji sustava. Razmišljanje o ključnim elementima problema identificira stvarne entitete koji postaju objekti unutar modela. Grupiranjem objekata vrši se klasifikacija temeljena na karakteristikama koja omogućuje razvrstavanje u razrede i definiranje nasljeđivanja. Opisivanjem veza među objektima teži se opisati načine na koje će objekt ispuniti zahtjeve, odnosno kako će se ponašati u odnosu na specifične zahtjeve. U konačnici, važno je opisati informacije koje svaki pojedini objekt mora sadržavati. To uključuje podatke i atribute koji su bitni za objekt te doprinose funkcionalnosti i svrsi objekta unutar sustava.

Pavlić (2011) navodi prednosti OO metodologije:

- Korisnici promatraju samo jedan model
- Ne odvajaju se podaci od procesa
- Razvojno osoblje obučeno u OO-u može biti produktivnije
- Modeli su jednostavniji
- Objekti se ponovo mogu koristiti

- Osoblje odjela informacijskih sustava lako uočava nedostatke
- OO metodologije temelje se na OO programskim jezicima poput Jave i C++-a

Prema istom autoru (Pavlić, 2011), nedostatci su sljedeći:

- „stariji“ informatičari vješti s SDLC-om teže prihvaćaju nove tehnologije
- Dodatna obuka za principe OO metodologije
- Potrebno je nabaviti alate za OO metodologiju
- Dodatni troškovi

3.3.1. UML

Prema Pavliću (2011), UML (eng. *Unified Modeling Language*) je osmišljen 1907. godine kada su Grady Booch, James Rumbaugh i Ivar Jacobson predali 1.0 verziju programa u *Object Management Group*. „UML je jezik za specijaliziranje, vizualiziranje, konstruiranje i dokumentiranje podataka softverskog sistema, kako za poslovna modeliranja tako i za druge nesoftverske sustave“ (Pavlić, 2011:277).

UML-ove prednosti u korištenju su sljedeće (Microsoft, 2019):

- Jednostavno predočava složene odnose
- Komunikacijske linije su otvorene
- Stvaranje softvera i procesa je automatizirano
- Pripomaže rješavanju teško uklonjivih problema s arhitekturom
- Povećava kvalitetu rada
- Smanjuje troškove i skraćuje ciklus izlaska na tržište

UML dijagrami dijele se na dvije glavne vrste, to su strukturni dijagrami i dijagrami ponašanja. Strukturni dijagrami prikazuju statičku strukturu softvera te sadrže različite razine apstrakcije i implementacije. Koriste se kako bi se vizualizirale strukture koje grade sustav, poput baze podataka te prikazuju hijerarhiju komponenti ili modula kao i način povezivanja i stupanja u interakciju. Dijagrami ponašanja su fokusirani na dinamične aspekte softverskog sustava te prikazuju funkcionalnost sustava i ističu što se mora napraviti u sustavu koji se modelira (Microsoft, 2019).

UML strukturni dijagrami su (Microsoft, 2019):

- Dijagram klasa

- Dijagram objekta
- Dijagram komponente
- Kompozitni strukturni dijagram
- Dijagram implementacije
- Dijagram paketa
- Dijagram profila

UML dijagrami ponašanja su (Microsoft, 2019):

- Dijagram slučajeva korištenja
- Dijagram aktivnosti
- Dijagram pregleda interakcija
- Dijagram tempiranja
- Sekvencijalni dijagram
- Dijagram komunikacije

U nastavku rada će biti ukratko objašnjena dva najčešće korištena UML dijagrama – dijagram klasa i dijagram slučajeva korištenja.

3.3.2. Dijagram klasa

Dijagram klasa (engl. *Class diagram*) je najčešći dijagram koji se koristi u razvoju softvera, a koristi se za opisivanje logičkog i fizičkog dizajna sustava Microsoft(2019). Ovaj dijagram daje vizualnu interpretaciju različitih klasa i načina na koji su one povezane. To je vrsta statičnog strukturnog dijagrama koji opisuje strukturu sistema tako da prikazuje sistemske klase, njezine atribute, operacije ili metode i veze između objekata. Dijagram klasa sadrži klase i relacije između klasa (Visual Paradigm, n.d.a).

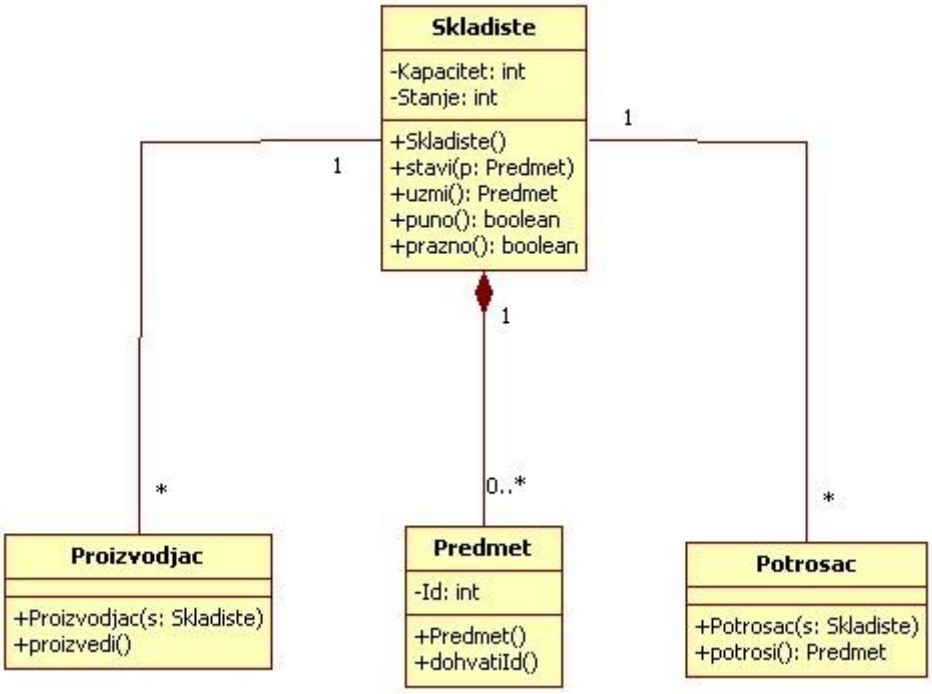
Visual paradigm (n.d.a) navodi svrhu dijagrama klasa:

- Pokazuje statičnu strukturu klasifikatora u sustavu
- Dijagram daje osnovnu notaciju za druge strukturne dijagrame propisane UML-om
- Pomaže programerima i drugim članovima tima
- Poslovni analitičari mogu koristiti dijagram klasa za modeliranje sustava iz poslovne perspektive

Avison i Fitzgerald (1995) definiraju klasu kao grupu objekata koji dijele uobičajene strukture i ponašanja. Klasa se sastoji od strukturnih značajka koje definiraju što objekti klase „znaju“ i ponašajnih značajka koje definiraju što objekt klase „može raditi“.

Bitno je spomenuti notaciju klase koja se sastoji od tri dijela. Ime klase, atribute i operacije. Atributi su svojstva klase, zapisuju se tzv. camelCase pravilom, što znači da se sve riječi pišu spojeno tako da prva riječ bude napisana malim slovom, a početno slovo druge riječi veliko (Open Network Foundation, 2015). Operacije su doslovne radnje koje obavlja objekt ili klasa. Veze koje postoje u dijagramu klase su generalizacija, kompozicija, agregacija, asocijacija i ovisnost. Svaka od veza ima svoju jačinu i svoj način prikazivanja (Medium, 2017).

Na slici 4 je prikazan primjer dijagrama klasa.



Slika 4. Primjer dijagrama klasa

Izvor: Information Technology School (ITS) (n.d.)

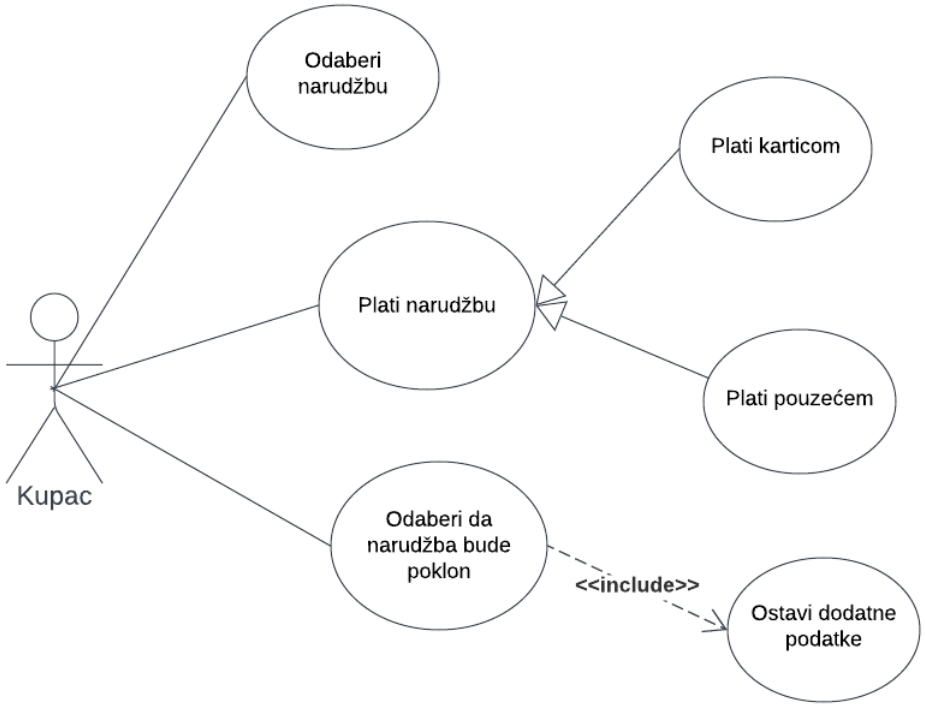
3.3.3. Dijagram slučajeva korištenja

Dijagram slučajeva korištenja (engl. *Use case diagram*) je jednostavni oblik UML dijagrama koji se većinski izrađuje u početnim fazama oblikovanja. Slučajevi korištenja određuju očekivano ponašanje, a ne točnu metodu ostvarivanja. Ključni koncept modeliranja je taj da

nam pokaže dizajnirati sustav iz perspektive krajnjeg korisnika. Prikazuje sva vanjska ponašanja i komunikacije sistema koja zahtjeva krajnji korisnik (Visual Paradigm, n.d.b).

Svrha korištenja dijagrama slučajeva korištenja je sljedeća (Visual Paradigm, n.d.b):

- Odrediti kontekst sustava
- Odrediti zahtjeve sustava
- Potvrditi sistemsku arhitekturu
- Potaknuti implementaciju i generirati testne slučajeve



Slika 5. Primjer dijagrama slučajeva korištenja
Izvor: Vlastita izrada (koristeći softver Lucidchart)

Slika 5 prikazuje jednostavan dijagram slučajeva korištenja u kojem kupac, koji predstavlja sudionika, obavlja online narudžbu nekog proizvoda. Kupac odabire što će naručiti, a zatim plaća narudžbu. Može birati hoće li platiti karticom ili pouzećem. Postoji opcija i da ta narudžba bude poklon, a u tom slučaju sudionik mora ostaviti dodatne podatke kako bi poduzeće od kojega naručuje znalo upakirati narudžbu. U ovom primjeru korištena je veza uključivanja

(engl. *include*) koja povezuje dva scenarija upotrebe, pri čemu jedan slučaj korištenja tijekom svog izvođenja u potpunosti izvodi drugi slučaj korištenja. Uz vezu uključivanja postoji još vrsta veza kao što su proširenje (engl. *Extend*), koja povezuje dva slučaja korištenja pri čemu jedan od njih proširuje funkcionalnost drugoga, veza generalizacije, u kojoj podređeni slučaj korištenja nasljeđuje ponašanja i značaj nadređenog slučaja korištenja (Visual Paradigm, n.d.b).

3.4. Agilne metodologije

Nema konsenzusa o preciznoj definiciji pojma „agilan“. Taj pojam izazvao je značajan interes među programerima, te u akademskoj zajednici. Radi se o metodologiji pomoću koje se razvija softver. Fokusira se na fleksibilnosti, suradnji i brzoj prilagodbi promjenama u zahtjevima i okruženju. Umjesto da se razvoj temelji na detaljnom i strogom planiranju unaprijed, agilna metodologija promiče iterativni i inkrementalni pristup, što znači da se timovi fokusiraju na isporuku vrijednih funkcionalnosti u kratkim vremenskim periodima, „iteracijama“ ili „sprintovima“. Agilna metodologija potiče bolju komunikaciju među članovima tima, provjere i prilagođavanja ciljeva, kontinuiranu integraciju i testiranje te brzu isporuku softverskih rješenja (Roić i Ferlež, 2020).

Razvoj Agilne metodologije najznačajnije raste 90-ih godina prošlog stoljeća, stvaranjem detaljno opisanih metodologija. Roić i Ferlež (2020) navode neke od najznačajnijih događaja vezanih uz razvoj i postanak agilne metodologije takve kakva je danas, a to su:

- 1995. godine Ken Schwaber i Jeff Sutherland izdaju rad „Scrum methodology“, što se smatra službenim početkom Scrum metodologije razvoja softvera.
- 1996. Tvrtka Rational izdaje Rational Unified Process (RUP)
- 1999. Kent Beck izdaju knjigu „Extreme Programming Explained“, ta metodologija je orijentirana prema razvojnim praksama i danas gotovo da nema projekta koji ne koristi barem neke njezine elemente

2001. godine izlazi objava koja je predstavila temeljne zaključke agilne metodologije. To su *„Agilni manifest“* i *„Načela agilnog razvoja“*. Osnovala ih je grupa *Agile Alliance* koja se sastojala od pojedinaca koji su težili promjenama u radu. Roić i Ferlež (2020) smatraju kako su „Agilni manifest“ i „Načela agilnog razvoja“ osnova za sav ostali sadržaj vezan uz agilne metodologije.

Agilni manifest se sastoji od četiri aspekta kojima se izlaže potreba za promjenom i poboljšanjem dotadašnjih načina rada i pristupa radu. Potiče se više cijeniti:

- Ljude i odnose, nego procese i alate;
- Upotrebljiv softver, nego iscrpnu dokumentaciju;
- Suradnju s korisnicima, nego pregovaranje oko ugovora;
- Reagiranje na promjenu, nego ustrajanje na planu

Načela agilnog razvoja prikazana su u Tablici 1.

Tablica 1. Načela agilnog razvoja

1	Naš prioritet je zadovoljstvo korisnika koje postižemo ranom i neprekinutom isporukom vrijednog softvera.	7	Upotrebljiv software osnovno je mjerilo napretka
2	Spremno prihvaćamo promjene zahtjeva, čak i u kasnoj fazi razvoja. Agilni procesi koriste promjene kako bi korisniku stvorili tržišnu prednost.	8	Agilni procesi podržavaju održivi razvoj. Pokrovitelji, razvojni inženjeri i korisnici trebali bi moći trajno zadržati jednak u brzinu.
3	Često isporučujemo upotrebljiv softver, u razmacima od nekoliko tjedana do nekoliko mjeseci, nastojeći da razmak bude čim manji.	9	Neprekinuta pažnja usmjerena tehničkoj izvrsnosti i dobar dizajn pospješuju Agilnost.
4	Poslovni ljudi i razvojni inženjeri moraju zajedno raditi svakodnevno, tijekom cjelokupnog trajanja projekta.	10	Jednostavnost – vještina maksimiziranja količine posla kojeg ne radimo – od suštinske je važnosti.
5	Projekte ostvarujemo oslanjajući se na motivirane pojedince. Pružamo im okruženje i podršku koja im je potrebna i prepuštamo im posao s povjerenjem.	11	Najbolje arhitekture, projektne zahtjeve i dizajn, stvaraju samoorganizirajući timovi.
6	Razgovor uživo najučinkovitiji je način prijenosa informacija razvojnome timu i međusobno unutar tima.	12	Timovi u redovitim razmacima razmatraju načine kako postati učinkovitiji, zatim se usklađuju i prilagođavaju svoje ponašanje.

Izvor: Samostalna izrada autora, prema Roiću i Ferležu (2020)

Agilna metodologija donosi razne prednosti, neke od njih su Ablison (n.d.):

- Kupcima je konstantno dostupan uvid u napredak projekta na kraju svake iteracije, što im omogućuje da prilagođavaju prioritete u skladu s vlastitim potrebama.
- Svaka iteracija omogućuje kupcima funkcionalni softver koji ispunjava njihova očekivanja, jer pruža stvarne rezultate nakon svakog sprinta.
- Velika prilagodljivost promjenama. Razvojni timovi su fleksibilni i mogu brzo reagirati na promjenjive zahtjeve. Prilagodba je moguća i u naprednim fazama razvoja.
- U agilnoj metodologiji prisutna je dvosmjerna komunikacija, tako da poslovni i tehnički dionici imaju jasnu vidljivost napretka projekta. Minimiziraju se nesporazumi i osigurava zajedničko razumijevanje ciljeva.
- Potiče se efikasan dizajn proizvoda koji udovoljava poslovnim zahtjevima.

Nedostatci agilne metodologije su Ablison (n.d.):

- Agilnom pristupu fali naglaska na dokumentaciju, jer taj nedostatak može dovesti do potencijalne pogrešne interpretacije agilnih timova. Gubi strogost i jasnoću vezanu za dokumentaciju.
- U nekim situacijama zahtjevi na početku mogu biti nedovoljno jasni, to može utjecati na promjenu vizije kupaca tijekom projekta, što zahtjeva implementaciju brojnih promjena. Timovi moraju biti svjesni potencijalnog nedostatka i biti spremni prilagodbi novim zahtjevima.

Postoji više vrsta agilnih metodologija, neke od najpoznatijih su Scrum, Kanban, Lean, Crystal, Ekstremno programiranje (XP), i druge. U nastavku će više biti razrađena Scrum agilna metodologija.

3.4.1. Scrum

Od svih agilnih metodologija najkorišteniji je okvir Scrum. Na tu činjenicu utječu faktori kao što su veličina područja koje pokriva ta metoda, kvalitetne prakse korištenja, ne bavi se tehnološkim, već organizacijskim praksama. Mnoge su prednosti Scruma koje će se u daljnjem tekstu spomenuti.

„Scrum je disciplinirani model razvoja softverskih proizvoda. Fokusiran je na rezultate i poslovnu vrijednost koja nastaje u projektu. Sav naglasak u procesu dan je na načela koja potiču vrijednost konačne isporuke, bez dodatnog i nepotrebnog rada“ (Roić i Ferlež, 2020:50). Bitno

je naglasiti da je Scrum vrlo jednostavan, lagan za koristiti i donosi mjerljive rezultate. Zbog tih karakteristika većinski je odabir u korištenju agilnih metodologija. Pokazalo se da korištenje Scruma potiče odgovornost korisnika, čini timove snažnijima a pojedince kvalitetnijim radnicima. Razlika Scruma i tradicionalnih metoda razvoja je povezivanje pojedinca s vrijednošću onoga što isporučuje. Roić i Ferlež (2020) navode kako Scrum potiče kreativnost na svim razinama, uz disciplinirane ključne razdiobe odgovornosti. Scrum se sastoji od tri dijela. To su 3 artefakta, 3 uloge i 4 ceremonije. Pomoću ove metode mogu se razvijati veliki i složeni procesi, no timovi ostaju mali jer tako zadržavaju međusobnu koheziju i povezanost.

Neke od prednosti Scruma i razloga zašto se tu metodu preferira više od drugih agilnih metodologija prema Roiću i Ferležu (2020) su:

- Iznimno je jednostavan za razumjeti
- Osnažuje tim i pojedince
- Povećava vidljivost dosegnutog posla za menadžment
- Scrum je mainstream, što znači da je lako pronaći suradnike koji ga poznaju
- Discipliniran je
- Smanjuje rizike od neuspjeha projekta

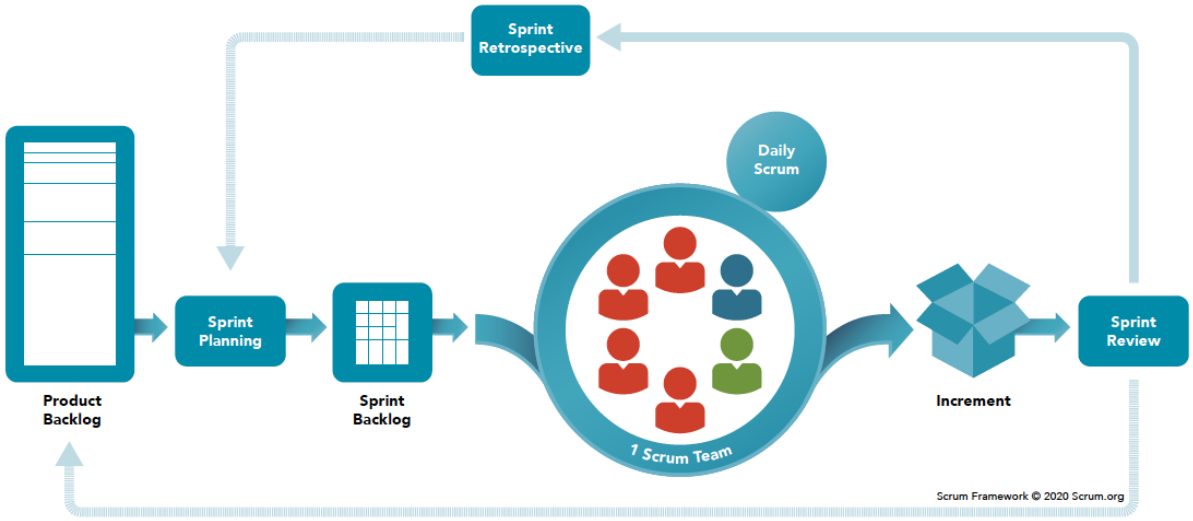
Ključan dio uvjeta korištenja Scruma bazira se na kontinuiranom surađivanju s timom kako bi se razjasnili i definirali zahtjevi. Korisnik je ključan sudionik koji kontinuirano stječe znanja o proizvodu, procesu razvoja softvera i vlastitoj ulozi. Korisnik donosi odluke u stvarnom vremenu i neprestano se konzultira s tehničkim i poslovnim stručnjacima (Roić i Ferlež, 2020).

Kao i sve metode, Scrum ima nedostatke, neki od njih su (Simplilearn, 2023):

- Često se smanjuje opseg zbog nedostatka vremena za rad
- Ukoliko pojedinci nisu potpuno predani i kooperativni povećavaju se šanse neuspjeha
- Potrebno je da svi članovi budu iskusni
- U tijeku izrade procesa trebaju biti prisutni svi članovi, u suprotnom kvaliteta projekta je narušena
- Kvalitetu je teško implementirati prije procesa testiranja

Scrum proces se sastoji od koraka rada. Ranije u radu su spomenuti sprintovi ili iteracije koje su sastavni dio svih agilnih metodologija. Prije svega definira se početna točka projekta u kojoj se nalazi popis zadataka koji se planiraju izvršiti. Prioritete zadataka zajednički određuju vlasnik proizvoda (engl. *Product Owner*) i klijent tako da uzimaju u obzir ravnotežu vrijednosti

i troškova. Tržište zahtijeva veliku kvalitetu i niske troškove, tako da stručnjaci moraju biti fleksibilni i agilni u razvoju proizvoda kako bi ostvarili krajnji ciljevi bez ugrožavanja kvalitete rezultata. Rezultati trebaju biti brzo gotovi. Slika 3 predstavlja vizualni prikaz procesa.



Slika 6. Scrum proces

Izvor: Scrum.org (n.d.)

4. Metodologija rada

Predmet istraživanja ovog završnog rada je pregled metodologija razvoja informacijskog sustava. Cilj rada je, na temelju relevantne stručne i znanstvene literature, napraviti usporednu analizu najzastupljenijih metodologija za razvoj informacijskih sustava.

Kriteriji prema kojima se radila analiza metoda su prije svega učinkovitost i kvaliteta izrade određenog zahtjeva. Još neki od kriterija su bili troškovi koje određena metoda snosi, rizici na koje se mora gledati prilikom planiranja rada te korisničko iskustvo i preferencije. Pomoću tih kriterija analiza je provedena tako da se sagledaju svi ali i procijeni koji od njih je za određeni zahtjev najpogodniji. Mnogi se autori prilikom analize metoda vode kriterijima kao što su specifičnost zadataka koji su dani, vrijeme trajanja određene metode i slično. Prilikom izrade ovog završnog rada kombinirani su kriteriji koje su autori smatrali bitnima i kriteriji koje preporučuju da se prate.

5. Usporedna analiza metoda razvoja informacijskih sustava

Nakon sažetog pregleda nekih od najčešćih metoda razvoja informacijskih sustava, napravit će se usporedna analiza istih. Usporedna analiza ovih metoda omogućuje razumijevanje njihovih prednosti i nedostataka u kontekstu razvoja informacijskih sustava. U ovom slučaju usporedbe analiza metoda razvoja informacijskih sustava kriteriji usporedbe su učinkovitost, kvaliteta izrade zahtjeva, troškovi koje nosi određena metoda i rizici. Prije usporedbe važno je napomenuti da se ne može odabrati jedna „najbolja“ metoda jer svaka nosi svoje prednosti i nedostatke, već je bitan kontekst koji nosi određeni posao.

Trenutno najzastupljenija metoda od svih navedenih je agilna metodologija. Neke od ključnih aspekta u kojima one predvode su iterativni i inkrementalni pristup. Učinkovitost u agilnim metodologijama znači brzu isporuku, prilagodljivost i suradnju tima. Agilne metodologije ističu se od drugih po tome što imaju brzu isporuku softverskih proizvoda, pomoću sprintova i iteracija olakšava rad i ubrzava dostavljanje gotovog proizvoda. Ovakvim načinom rada potiče se suradnja s kolegama i reagiranje na promjene više nego pregovaranje oko ugovora i ustrajanje na planove (Roić i Ferlež, 2020). Učinkovitost se također očituje u tome što se više bazira na konkretan rad nego na dokumentaciju, no to u nekim slučajevima može biti negativno ukoliko je dokumentacija ključna. Kao što je već navedeno, faze razvoja se provode u sprintovima ili iteracijama koje omogućuju fokusiranje na jedan segment procesa. Svaka od iteracija donosi funkcionalnosti i vrijednosti korisnicima, što omogućuje brže dobivanje povratnih informacija te prilagođenost na promjene. Promatrajući kriterij kvalitete izrade zahtjeva, funkcionalni dijelovi proizvoda u agilnim metodologijama su kontinuirano testirane i verificirane. Također, fleksibilnost u prilagodbi zahtjeva je jedna od dobrih karakteristika kriterija kvalitete izrade zahtjeva. Troškovi koji se pojavljuju mogu varirati, a oni se većinom odnose na obuku tima, alate, tehnologije i slične troškove. Mogući rizici su nepotpunost projekta zbog vremenske ograničenosti, neskladnost tima.

Uzevši za primjer strukturnu metodu, prilikom razvoja informacijskog sustava teže se prilagođava na promjene jer je statička metoda te bi mijenjanje jednog segmenta narušilo funkcionalnost ostalih. Učinkovitost u ovim metodologijama je popraćena dokumentacijom i predvidljivošću. Kao što je u prethodnim poglavljima opisano, strukturne metodologije imaju kontrolu nad razvojnim procesom. Učinkovitost može biti povećana zbog toga što se omogućava upravljanje rizicima, promjenama i kvalitetom tijekom cijelog procesa. Dokumentacija omogućava kontinuirani uvid u sve korake nekog projekta, tako da rad može biti jasniji.

U životnom ciklusu razvoja softvera, koja je najstarija metodologija, procesi se odvijaju sporije. Razlog tome su faze razvoja koje se rade jedna za drugom, kontroliraju se promjene tijekom razvoja softvera, pruža stabilnost i sigurnost u izradi, te detaljno planira svaku fazu razvoja procesa. Troškovi ove metodologije također mogu varirati, potrebni su resursi, alati, infrastrukture, testiranja i drugi. Rizici se mogu pojaviti prilikom lošeg planiranja, nedovoljne kvalitete, lošeg upravljanja, promjena zahtijeva i sličnih komponenti. Tu se vidi razlika u rizicima u odnosu na druge metode jer se većinski baziraju na načinu organiziranja i vođenja posla.

Objektno-orijentirana metoda je jedna od najjednostavnijih metoda za rad, ali prikazuje procese koji su predvidivi na samom početku procesa. Dijagrami koje nudi se još zovi i „prvi dijagrami“, što ukazuje na površnost njenih procesa. U drugim metodama, projekt se promatra s puno šireg aspekta. Objektno-orijentirana metoda može poboljšati učinkovitost razvojnog procesa kroz par svojih obilježja. Neka od njih su ta da se jednom napravljeni objekti mogu koristiti ponovno prilikom izgradnje novog procesa, smanjuje se vrijeme i napor izrade potpuno novog procesa. Struktura je jasna i dobro organizirana, što ujedno i olakšava suradnju tima tako da svaki pojedinac može raditi svoj dio neovisno o drugima. Uz ove učinkovitosti mogu se javiti i rizici kao što su loš dizajn, loša implementacija, nedostatak jasnoće te rizik loše definiranih odnosa između objekata. Troškovi mogu nastati kod obuke novih djelatnika jer je metoda novija, resursi, alati, instalacije, testiranja i drugi troškovi. Kvaliteta ove metode očituje se u alatima koji omogućuju suvremeni i jednostavan način prikaza procesa, što ujedno olakšava krajnjim korisnicima i timu shvaćanje problema zadatka.

Gledano sa strane stručnjaka koji rade na razvoju može se usporediti lakoća obavljanja posla, opseg posla i snalaženje s alatima. Agilne metode se razvijaju u timskom okruženju, što zahtjeva kontinuiranu suradnju s korisnicima. Jače je naglašen timski rad nego u drugim metodama. Objektno-orijentirane metode su lakše za korištenje jer se baziraju na izradi dijagrama. Alati su dostupni i jednostavni za korištenje. Strukturne metode zahtijevaju duže vrijeme izrade i planiranja procesa.

Dvije analize koje koriste dijagrame kao glavne alate su objektno-orijentirana i strukturna metoda. Objektno-orijentirana analiza dijagrame koristi kako bi modelirala objekte, njihove atribute, ponašanja i međuovisnosti. Ovi dijagrami pružaju uvid u strukturu i dinamiku objekata, identificiraju nasljeđivanje, asocijacije i drugo. S druge strane, strukturna analiza koristi dijagrame kako bi se prikazala struktura i organizacija cijelog sustava. Oni pomažu u

analizi funkcionalnosti sustava, identifikaciji ulaza, izlaza i procese te definiranju logičke organizacije sustava.

6. Rasprava

Ovim radom prikazuju se ključne komponente metoda razvoja informacijskih sustava i njihova analiza. Analizirane su sljedeće metodologije: životni ciklus razvoja softvera, strukturna metoda, objektno-orijentirane metode i agilne metodologije.

Životni ciklus razvoja softvera sastoji se od strogo definiranih faza te osigurava korake koji se moraju provesti kako bi se postigao željeni cilj, odnosno isporučio funkcionalan proizvod. Poduzeća mogu koristiti SDLC metodologiju kada imaju složene i dugotrajne projekte koji zahtijevaju detaljno planiranje, precizne zahtjeve i strogo upravljanje promjenama. Za primjer bi se moglo uzeti zdravstvo kao industrija u kojoj korištenje SDLC metodologije donosi najbolje rezultate.

S druge strane, strukturna metoda se temelji na jasno definiranoj hijerarhiji i strukturi sustava. U ovom pristupu se naglašava postizanje stabilnosti i dobre organizacije. Također postoje ograničenja koja se javljaju prilikom prilagodbe promjenama zahtjeva. Karakteristike ove metode mogu se primjenjivati u industrijama koje imaju stroge regulative i zahtjeve.

Objektno-orijentirane metode se temelje na identifikaciji objekata i njihovim međuodnosima. Ona olakšava održavanje sustava. Sve industrije koje koriste informacijske tehnologije mogu koristiti objektno-orijentirane metode zbog toga što olakšavaju upravljanje i održavanje modeliranje i dizajniranje složenih sustava.

Na kraju, agilne metodologije, kao najpopularniji izbor metoda, su istaknute zbog svoje fleksibilnosti i suradnje tima. Potiču brze iteracije, komunikaciju i stavljaju fokus na isporuku vrijednosti klijentu. Industrije za koje je karakteristično koristiti agilne metodologije su one koje rade u dinamičnim i inovativnim sektorima, kao što su razvoji mobilnih i web sadržaja, te digitalni marketing. U takvim industrijama treba se brzo reagirati i brzo dati rješenje.

Svaka od ovih metoda se ne može univerzalno smatrati najboljom za sve projekte i organizacije. Uzimajući u obzir sve analizirane metode, može se zaključiti da je ključno razumjeti kontekst projekta i potrebe organizacije i pri odabiru metode temeljito pregledati karakteristike, rizike i prednosti metoda. Ukoliko se odabere kriva metoda i krene po njoj razvijati informacijski sustav, nastat će problemi koje će biti teže rješavati, nego probleme koji su nastali u okruženju razvoja s odgovarajućom metodom i proces razvoja bi se trebao raditi ispočetka. Kako metode, tako i korisnici moraju uzeti u obzir otvorenosti za prilagodbu i fleksibilnost prilikom odabira metode razvoja informacijskih sustava.

7. Zaključak

U ovom završnom radu provedena je analiza par značajnijih metoda razvoja informacijskih sustava. Te metode su životni ciklus, strukturne metode, objektno-orijentirane metode i agilne metodologije. Cilj završnog rada bio je identificirati prednosti, nedostatke i primjenu svake od metoda te pružiti relevantne smjernice koje pomažu pri odabiru odgovarajuće metode. Analizom podataka o pojedinoj metodi dan je uvid u to kako funkcionira pojedina metoda i koje su njene prednosti i nedostatci. Zaključak je da svaka metoda može najbolje pridonijeti razvoju određenog sustava ako se dobro sagledaju zahtjevi sustava i kontekst izrade.

Za kraj važno je napomenuti da svaka od faza pogoduje određenom tipu zahtjeva. Ne može se gledati općenito koja je najbolja ili najlošija, već je potrebno promatrati kontekst svakog sistemskog zahtjeva. Tako, na primjer, ako se kreće u razvoj informacijskog sustava, a unaprijed se zna da bi prilikom razvoja moglo doći do određenih promjena, neće se odabrati strukturna metoda ili SDLC, već neka od agilnih metodologija. Ako zahtjev podrazumijeva izradu jednostavnog modela koji će dati šturi prikaz radnji, odabrat će se neka od objektno-orijentiranih metoda.

U konačnici, razvoj informacijskih sustava je vrlo širok i opsežan, no ima mnogo mjesta za buduće proširenje analiza i usporedba. Tehnološki napredak i promjene u industriji će mnogo utjecati na relevantnost i primjenu ovih metoda u budućnosti. To ostavlja mjesta za daljnji razvoj i produbljenje ovih analiza, možda i stvaranje nove jedinstvene metode koja će biti prigodna za više različitih načina razvoja.

Literatura

1. Ablison (n.d.) Pros And Cons Of Structured Analysis. Dostupno na: https://www.ablison.com/pros-and-cons-of-structured-analysis/?expand_article=1 [pristupljeno 10.7.2023.]
2. Avison D. E., Fitzgerald G. (1995). *Information System Development: Methodologies, Techniques and Tools*. McGraw-Hill Book Company Europe
3. Eternal sunshine of IS mind (n.d.) Context Level DFD's & Level 1 DFD's. Dostupno na: https://eternalsunshineoftheismind.wordpress.com/2013/03/05/context-level-dfds-level-1-dfds/ [pristupljeno 25.7.2023.]
4. Garača, Ž. (2008). *Poslovni informacijski sustavi.* Sveučilište u Splitu, Ekonomski fakultet
5. Information Technology School (ITS) (n.d.) UML – Sistem za skladištenje predmeta. Dostupno na: https://www.its.edu.rs/uml-sistem-za-skladiste-predmeta/ [pristupljeno 25.7.2023.]
6. Link-eLearning (n.d.) Softverski dizajn. Dostupno na: https://www.link-elearning.com/site/kursevi/lekcija/4332 [pristupljeno 15.7.2023.]
7. Medium (2017). UML Class Diagrams Tutorial, Step by Step. Dostupno na: https://medium.com/@smagid_allThings/uml-class-diagrams-tutorial-step-by-step-520fd83b300b [pristupljeno 23.6.2023.]
8. Microsoft (2019). Jednostavan vodič za izradu UML dijagrama i modeliranja baza podataka. Dostupno na: https://www.microsoft.com/hr-hr/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling [pristupljeno 23.6.2023.]
9. Open Networking Foundation (2015). UML Modeling Guidelines. Dostupno na: https://opennetworking.org/wp-content/uploads/2014/10/UML_Modeling_Guidelines_V1.0.pdf [pristupljeno 10.7.2023.]
10. Pavlić, M. (2011). *Informacijski sustavi.* Školska knjiga
11. Roić, R., Ferlež L. (2020). *Agilni razvoj softvera.* AG04 Innovative Solutions d.o.o. Zagreb

12. Scrum.org (n.d.) What is Scrum? Dostupno na: https://www.scrum.org/resources/what-scrum-module [pristupljeno 25.7.2023.]
13. Simplilearn (2023). Scrum Project Management: Advantages and Disadvantages. Dostpuno na: https://www.simplilearn.com/scrum-project-management-article [pristupljeno 10.7.2023.]
14. Techopedia (2016.) Structured Analysis https://www.techopedia.com/definition/24637/structured-analysis [pristupljeno 20.6.2023.]
15. Tutorialspoint (n.d.) Structured Analysis https://www.tutorialspoint.com/system_analysis_and_design/system_analysis_and_design_structured.htm# [pristupljeno 20.6.2023.]
16. Visual Paradigma (n.d.a.) What is Class Diagram? Dostupno na: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/ [pristupljeno 10.7.2023.]
17. Visual Paradigma (n.d.b.) What is Use Case Diagram? Dostupno na: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ [pristupljeno 22.6.2023.]

Popis slika

Popis tablica