JEDRO IN JEZIK ESSENCE V KONTEKSTU KAKOVOSTI PROGRAMSKE OPREME

ROBERT LESKOVAR, ZVONKO BELIČ
Univerza v Mariboru, Fakulteta za organizacijske vede, Kranj, Slovenija
robert.leskovar@um.si, zvonko.belic@student.um.si

Sinopsis Raziskava naslavlja priložnosti povezovanja področij standardov kakovosti programske opreme in inženiringa programske opreme s pristopom Essence. Velik izziv v disciplini programskega inženiringa je pomanjkanje splošno uveljavljene metode, ki rezultira v visoko kakovostni programski opremi. Metoda bi morala biti uporabna in prilagodljiva ter preprečevati ujetost v dogmo specifične metodologije. Predstavljen je standard kakovosti programske opreme ISO/IEC 25010, ki opredeljuje atribute oz. kriterije kakovosti izdelka (funkcijska stabilnost, učinkovitost delovanja, združljivost, uporabnost, zanesljivost, varnost, vzdrževalnost, prenosljivost) in kriterije kakovosti v uporabi (uspešnost, učinkovitost, zadovoljstvo, odsotnost tveganj). Sledi analiza jedra in jezika za razvoj metod v programskem inženirstvu z nazivom Essence. Nakazuje se priložnost, da Essence povežemo z modelom kakovosti programske opreme tako, da kriterije kakovosti izdelka preslikamo v alfo Zahtevane lastnosti, kriterije kakovosti v uporabi pa v alfo Priložnosti. Na ta način premostimo prepad med potrebo po stabilnosti metode izgradnje informacijskega sistema in potrebo po agilnem prilagajanju spremembam.

Ključne besede:
programska oprema, kakovost programske opreme, metode programskega inženiringa, Essence, informacijski sistem

Univerzitetna založba Univerze v Mariboru

DOI https://doi.org/10.18690/um.fov.6.2022.10
ISBN 978-961-286-637-2

KERNEL AND ESSENCE LANGUAGE IN THE CONTEXT OF SOFTWARE QUALITY

ROBERT LESKOVAR, ZVONKO BELIČ
University of Maribor, Faculty of Organizational Sciences, Kranj, Slovenia
robert.leskovar@um.si, zvonko.belic@student.um.si

Abstract This research addresses the opportunities of linking software quality standards and software engineering with the Essence approach. A major challenge in the discipline of software engineering is the lack of a generally accepted method, resulting in high quality software. The method should still be useful, flexible and avoid getting caught up in the dogma of a specific methodology. The ISO/IEC 25010 software quality standard is presented, which defines the quality attributes or criteria (functional stability, performance efficiency, compatibility, usability, reliability, security, maintainability, portability) and quality criteria in use (effectiveness, efficiency, satisfaction, freedom of risks, content completeness). An analysis of the Essence kernel and language for a method development is given. There is an opportunity to connect Essence with software quality model in a way that product criteria are mapped into Alpha Requirements, and the criteria in use are mapped into Alpha Opportunities. This way, we bridge the gap between the need for development method stability and the need for agile adaptation to changes in the system.

Keywords: software, software quality, software engineering, methods, Essence information system

https://doi.org/10.18690/um.fov.6.2022.10 DOI
978-961-286-637-2 ISBN
University of Maribor Press

1 Uvod

Kakovost programske opreme je po definiciji standarda ISO/IEC 25010:2017 Software Quality Model (ISO/IEC, 2017) stopnja, do katere sistem zadovoljuje izražene in samo po sebi umevne potrebe različnih deležnikov ter jim zagotavlja vrednost. Ta standard kakovost programske opreme razdeli v dva široka segmenta: kakovost izdelka in kakovost v uporabi. Izdelek je kakovosten v uporabi, če je bil predhodno izveden uspešen projekt razvoja. Merjenje uspešnosti razvoja programske opreme se pri projektih najpogosteje omeji na trojico {čas, stroški, obseg}, merjenje učinkovitosti pa na razmerje med rezultati projekta (npr. število vrstic kode, število funkcijskih točk) in porabljenimi viri (npr. vloženi čas, število človek–mesec, stroški v denarnih enotah). V literaturi s področja kakovosti programske opreme zasledimo tudi t. i. projektni diamant, ki vključuje {čas, kakovost, obseg in stroške}, npr. (Akbar et al., 2017). Uspešen projekt razvoja programske opreme se torej v veliki meri sklada s kakovostjo razvite programske opreme.

Vprašanje uspešnosti projektov razvoja programske opreme je v disciplini programskega inženiringa poznano več desetletij. Reel (1999) v znamenitem članku »Critical Success Factors in Software Projects« navaja povečanje deleža propadlih projektov ter deset glavnih razlogov: vodje projektov ne razumejo uporabnikovih potreb, obseg projekta je neustrezno definiran, spremembe projekta so slabo vodene, izbrana tehnologija se v teku projekta spremeni, poslovne potrebe se spremenijo, roki za izdelavo so nerealistični, uporabniki se upirajo rešitvi, sponzor projekta izgine, pomanjkanje kompetenc v razvojnem timu in vodje projektov ignorirajo dobre prakse. Med temi razlogi jih je sedem, ki so zaznavni v trenutku, ko ni napisana niti ena sama vrstica programske kode. Čeprav je od tega zapisa minilo več kot dvajset let, problem neuspešnih projektov, v katere je vloženo ogromno virov, ni nič manjši.

Članek Dendere (Dendere et al., 2021), ki se sklicuje na zapise v medijih in strokovno literaturo, navaja globalni trend neuspešnih projektov v zdravstvu, povezanih z digitalno transformacijo. Glavni vzrok naj bi bilo vodenje projektov, bolj natančno: izbira primernega pristopa za vodenje projekta (»Adopting a suitable project management approach is a major factor for achieving success because managing a project using an unsuitable methodology can severely damage the

chances of success.«). Članek implicitno favorizira agilne pristope pred tradicionalnimi, vendar brez empiričnih dokazov.

Rasheed (Rasheed et. al., 2021) postavlja večjo uspešnost programskih projektov, ki so uporabljali agilni način razvoja, v nekoliko drugačno luč. Avtorji trdijo, da je visok delež uspešnih projektov razvoja programske opreme z agilnimi metodami bolj posledica velikosti projektov (oz. bolje majhnosti). Navedeni so tudi primeri uspešne uporabe agilnega pristopa v velikih, kompleksnih projektih. Izpostavljajo uporabo agilne metodologije SAFe (Scaled Agile Framework). Inženiring zahtev (requirements engineering) je po njihovem mnenju ključni del življenjskega cikla projekta, čeprav navajajo, da vzroka za kar 50 % neuspešnih projektov niso uspeli identificirati. Kot identificirane vzroke neuspeha navajajo pomanjkanje tehničnih kompetenc (13 %), nepopolne zahteve (12 %), spreminjajoče se zahteve (12 %), slabo sodelovanje uporabnikov (7 %) in slab začetek projekta (6 %).

V preteklih desetletjih se je virtualni boj za prevlado metodologij razvoja vedno bil med tradicionalnim pristopom na osnovi življenjskega cikla in »novitetami«, kot so agilni razvoj, SCRUM, XP, FDD, TDD in SAFe. Toda številne študije so pokazale, da je praksa, tj. način dela razvijalcev najpomembnejši za uspeh projekta. Znano je, da organizacije, ki dosežejo višjo stopnjo zrelosti procesa razvoja (Chaudhary & Chopra, 2017), pogosteje dokončajo projekt na zadovoljstvo naročnika, v predvidenem času in s predvidenimi viri. Dobre prakse so vključene tudi v številne standarde s področja programskega inženirstva, npr. opredelitev zahtevanih lastnosti v standardu ISO/IEC 29148:2018 (ISO/IEC/IEEE, 2018) ali pa ISO/IEC 25010:2017 (ISO/IEC, 2017), ki opredeljuje atribute kakovosti izdelka in kakovosti v uporabi. Vedno znova se je izkazalo, da disciplina programskega inženirstva potrebuje splošno, a prilagodljivo in sprejeto metodologijo, ki ne sme sama po sebi omejevati razvijalcev (per-se), temveč jim mora omogočati učinkovito in uspešno delo. V tem prispevku bomo zato obravnavali pristop, ki omogoča gradnjo metodologije razvoja programske opreme – Essence. Jezik in jedro Essence izvirata iz iniciative SEMAT, ki je povezala skupino strokovnjakov v prizadevanju utemeljiti področje programskega inženirstva kot rigorozno znanstveno disciplino. Mednarodni, neprofitni konzorcij OMG (Object Management Group) je na osnovi njihovega dela izdelal standard, ki je definiral skupne elemente, jezik in okvir za izdelavo metod v programskem inženirstvu. OMG je leta 2014 izdal verzijo 1.0, leta

2015 verzijo 1.1 in leta 2018 verzijo 1.2 standarda za jezik in jedro Essence (Park et al., 2018).

Pregled literature in stanje v praksi nakazujeta, da med sfero mednarodnih standardov in metodologij razvoja obstaja prepad, ki je podoben tistemu med oddelkom razvoja in oddelkom, ki izvaja operacije. Na eni strani gre za potrebo po stabilnosti, na drugi pa zaradi spreminjajočih se okoliščin nuja po novostih. V nadaljevanju prispevka je najprej predstavljen segment kakovosti programske opreme, opredeljen z vidika atributov kakovosti. Ta predstavlja agregat dobrih praks iz preteklosti in nudi stabilno oporo. Sledi predstavitev značilnosti jezika in jedra Essence kot orodja za izgradnjo novih metod in praks. Ta del omogoča vpeljavo najsodobnejših pristopov in tehnologij. V diskusiji pa so nakazane priložnosti, ki izhajajo iz povezave gradnje novih metod z Essence in področja kakovosti programske opreme.

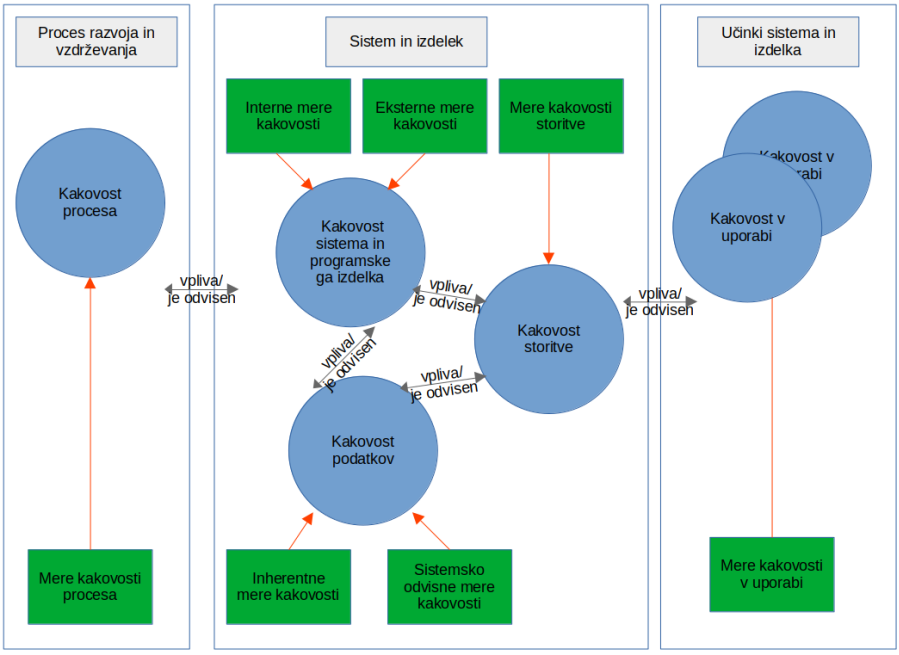
2 Izhodišča kakovosti programske opreme

V razvoju področja kakovosti programske opreme so se v preteklosti pojavljali številni modeli, ki so bili osnova za današnje standarde. McCallov faktorski model (McCall et. al., 1997) definira tri kategorije karakteristik ali kriterijev kakovosti: a) izvajanje {pravilnost, zanesljivost, učinkovitost, integriteta in uporabnost}, b) vzdrževanje {vzdrževalnost, fleksibilnost in testabilnost} ter c) prenos {prenosljivost, ponovna uporaba in interoperabilnost}. Boehm (Boehm et. al., 1978) je še izboljšal McCallov model s tem, da je definiral funkcijo koristi kot trojico {priročnost, prenosljivost in vzdrževalnost}. Dromey (Dromey, 1995) je v svojem modelu izhajal iz zanesljivosti in vzdrževalnosti. Standard ISO 9126 (ISO, 1991 in 2001) je definiral in agregiral prej naštete karakteristike kakovosti programske opreme. Leta 2011 ga je nadomestil trenutno veljavni standard ISO/IEC 25010:2017 (ISO/IEC, 2017). Ta standard izhaja iz treh bistvenih gradnikov:

- **proces**: kakovost procesa vpliva na interne lastnosti razvite programske opreme. Kakovost procesa se meri s procesnimi merami;
- **programska oprema**: standard ločuje interne in eksterne lastnosti. Interne lastnosti vplivajo na eksterne lastnosti, te pa z uporabo povzročajo učinke. Interne lastnosti so odvisne od kakovosti procesa, eksterne lastnosti pa od internih;

– **učinki**: učinke predstavlja »kakovost v uporabi«. Standard definira mere »kakovosti v uporabi«. Kakovost v uporabi je odvisna od kakovosti programske opreme in od konteksta uporabe.

Povezanost navedenih gradnikov prikazuje slika 1.

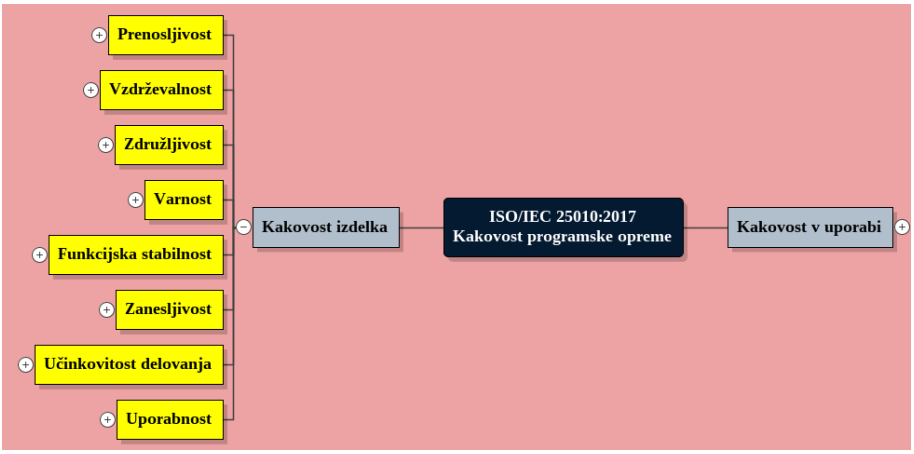


Slika 1: Povezave in odvisnosti med merami kakovosti po standardu ISO/IEC 25010:2017
Vir: lasten.

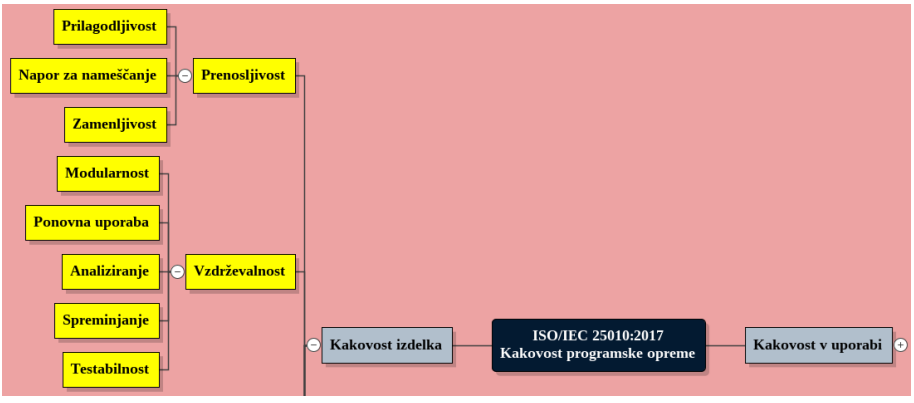
Model kakovosti programske opreme po standardu ISO/IEC 25010:2017 sestavljata kakovost izdelka in kakovost v uporabi.

Kakovost izdelka opredeljujejo naslednji kriteriji: funkcijska stabilnost (do kolikšne mere je izdelek skladen z definiranimi in samo po sebi umevnimi potrebami v specificiranih pogojih), učinkovitost delovanja (nanaša se na porabo virov v specificiranih pogojih), združljivost (stopnja, do katere izdelek izmenjuje informacije z drugimi izdelki ob tem, da izvaja zahtevane funkcije), uporabnost (stopnja, do katere specificirani uporabniki v specificiranem kontekstu dosegajo specificirane cilje učinkovito, uspešno in z zadovoljstvom; ta kriterij je lahko upoštevan tako pri izdelku

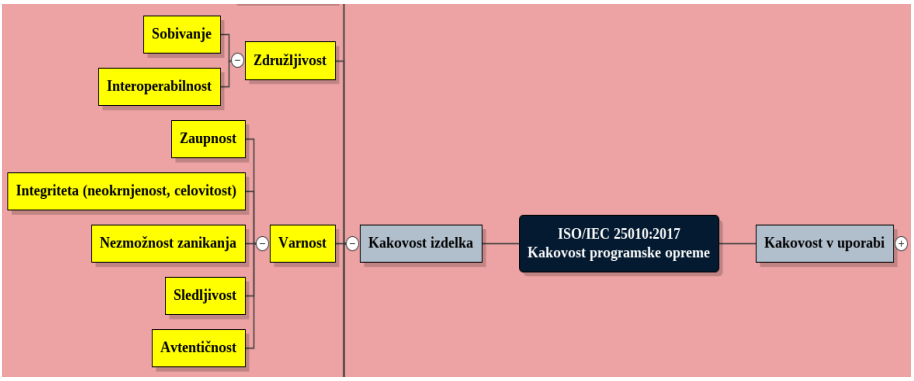
kot v uporabi), zanesljivost (stopnja, do katere izdelek izvaja specificirane funkcije v specificiranih pogojih specificirano obdobje), varnost (v kolikšni meri so podatki varni, vključno s pravicami dostopa), vzdrževalnost (stopnja, do katere je možno izdelek spreminjati uspešno in učinkovito), prenosljivost (mera učinkovitosti in uspešnosti prenosa iz enega delovnega okolja v drugo). Slike od 2 do 6 prikazujejo drevo kriterijev kakovosti izdelka.



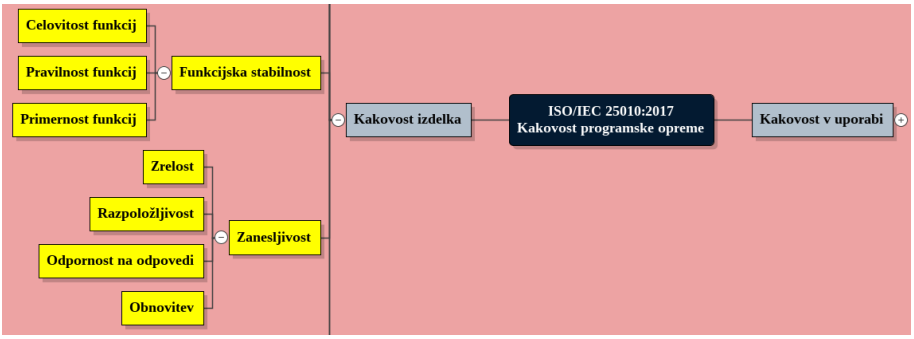
Slika 2: ISO/IEC 25010 – glavni kriteriji kakovosti izdelka
Vir: lasten.



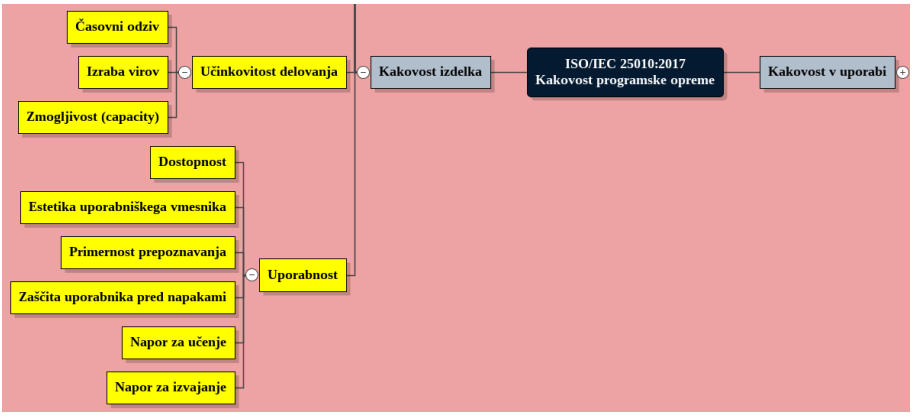
Slika 3: ISO/IEC 25010 – kriterija kakovosti izdelka: prenosljivost in vzdrževalnost
Vir: lasten.



Slika 4: ISO/IEC 25010 – kriterija kakovosti izdelka: združljivost in varnost
Vir: lasten.

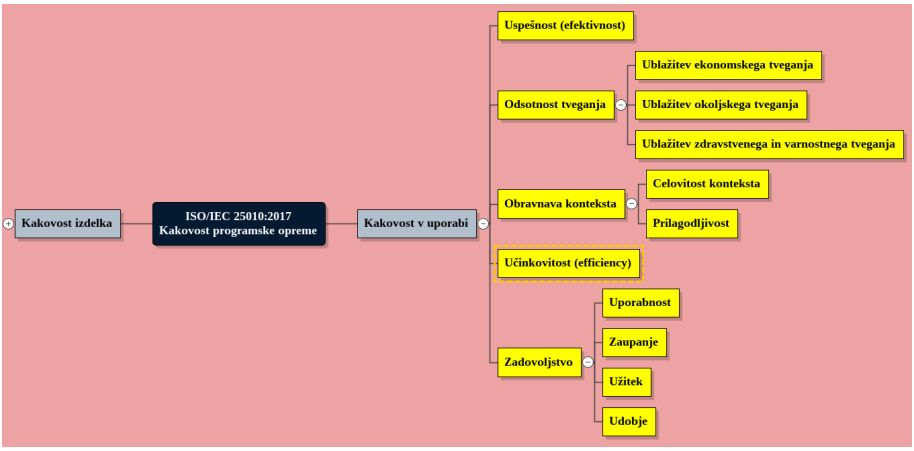


Slika 5: ISO/IEC 25010 – kriterija kakovosti izdelka: funkcijska stabilnost in zanesljivost
Vir: lasten.



Slika 6: ISO/IEC 25010 – kriterija kakovosti izdelka: učinkovitost delovanja in uporabnost
Vir: lasten.

Kakovost v uporabi definira pet glavnih kriterijev, od katerih imajo trije še pridružene podkriterije: uspešnost (predstavlja stopnjo natančnosti in popolnosti, do katere uporabniki dosegajo specificirane cilje), učinkovitost (poraba virov, ki je povezana z določeno stopnjo uspešnosti), zadovoljstvo (stopnja, do katere so potrebe uporabnika zadovoljene v specificiranem kontekstu uporabe izdelka; v to so vključena tudi pričakovanja uporabnikov), odsotnost tveganj (to vključuje denarna, okoljska in zdravstvena tveganja) in obravnava konteksta (celovitost in prilagodljivost). Na sliki 7 so prikazani kriteriji kakovosti v uporabi.



Slika 7: ISO/IEC 25010 – kriteriji in podkriteriji kakovosti v uporabi
Vir: lasten.

Standard je z leti postal celovitejši, a po drugi strani bolj nepregleden. Moteče je, da vsebinsko različnim kriterijem dovoljuje enak naziv – uporabnost. V enem primeru se uporabnost nanaša na stopnjo doseganja nečesa, v drugem primeru pa se nanaša na napor, ki je potreben za doseganje cilja uporabnika. Zato bo zelo verjetno prišlo do sprememb standarda ravno pri uporabnosti.

Po mnenju avtorjev tega prispevka bi bil najbližji nadomestni izraz uporabniška izkušnja, s tem, da ima angleški izraz »user experience« težavo, saj ne zajame tega, kar z dvema različnima besedama povemo npr. v slovenskem ali nemškem jeziku. Ta izraz bi se nanašal tako na kratkotrajne učinke v smislu uporabniškega doživetja (nemško Erlebniss), torej na emocionalno doživljanje uporabe, in po drugi strani na dalj časa trajajoče učinke v smislu izkušnje (nemško Erfahrung), torej racionalni, kognitivni vidik uporabe. Morda bi bila v angleškem jeziku primernejša rešitev s

stopnjevanjem npr. »short-term user experience« in »long-term user experience« kot dosedanji kriteriji zadovoljstva {uporabnost, zaupanje, užitek, udobje}. Posebej zadnji trije so problematični, saj v času uporabe stopnja zaupanja, užitka in udobja lahko zelo nihajo. Časovna določitev pa bi odpravila to divergenco stanj. Predlog sprememb kriterija kakovosti »zadovoljstvo« je odstranitev dosedanjih podkriterijev {uporabnost, zaupanje, užitek, udobje}, nadomestitev kriterija zadovoljstvo z izrazom »uporabniška izkušnja (angl. »user experience«, nem. »Benutzererfahrung«) ter določitev dveh podkriterijev: a) »doživetje« (angl. »short-term user experience«, nem. »Erlebnis«) in b) izkustvo (angl. »long-term user experience«, nem. »Erfahrung«). Gostota doživetij in izkušenj v specificiranem času bi bila bolj natančna in bolj merljiva mera kakovosti, kot so dosedanje.

3 Jedro in jezik Essence

3.1 Jedro

Essence izvira iz iniciative SEMAT v letu 2009 (Jacobson et al., 2019). Object Management Group je na osnovi iniciative SEMAT začela z razvojem standarda za jezik in jedro Essence ter leta 2014 izdala verzijo 1.0. Najnovejša verzija 1.2 je bila izdana leta 2018 (Object Management Group, 2018). Tako SEMAT kot Essence izhajata iz problematike discipline programskega inženirstva – iskanje zrelih praks razvoja programske opreme, ki rezultirajo v uspešnih projektih in kakovostnem izdelku. Jacobson (Jacobson et al., 2019) ocenjuje, da je v svetovnem merilu okoli 20 milijonov razvijalcev in preko 100 tisoč različnih metod. Skoraj vsaka skupina razvijalcev dela na svoj način. Nove metode se nenehno pojavljajo. Eden od takih primerov izboljševanja procesa izdelave in posledično kakovosti programske opreme je Model A-Z (Akbar et al., 2017). Temelji na pristopu SDLC (tradicionalni), v katerega vpelje dodatne aktivnosti za povečanje kakovosti izdelka. Avtorji trdijo, da natančnejša časovna opredelitev aktivnosti (time-boxing) za vsako fazo življenjskega cikla ob pogoju, da je to mogoče, poveča možnosti za uspešen projekt in kakovosten izdelek. Najpomembnejša lastnost jedra in jezika Essence je neodvisnost od pristopa, metodologije oz. metode. V poglavju 4 Essence (Object Management Group, 2018) definira naslednje ključne termine:

- Aktivnost (Activity): ena ali več vrst dela z navodili, kako delo opraviti;

- Prostor aktivnosti (Activity space): oznaka za nekaj, kar mora biti opravljeno v sklopu prizadevanja. Oznaka lahko obsega eno ali več aktivnosti;
- Alpha ali alfa (Alpha): Alfa je bistveni element prizadevanja programskega inženiringa, ki je pomemben za oceno napredka in »zdravja« prizadevanja. Alpha je akronim za »Abstract-Level Progress Health Attribute«. V slovenščini bi to lahko dobesedno prevedli kot »atribut zdravja abstraktnega nivoja napredka«, vendar bi izgubili asociacijo na nekaj, kar je najpomembnejše (alfa). Zato bomo v nadaljnjem besedilu ohranili izvirni akronim, kadar bo to sprejemljivo z jezikovnega vidika, sicer bomo uporabljali sinonim »alfa«;
- Povezava med Alphami (Alpha association): povezava med dvema bistvenima elementoma (Alpha-mi);
- Področje pozornosti (Area of concern): elementi jedra ali praktičnih pristopov so lahko razdeljeni na skupine področij, ki zahtevajo pozornost v prizadevanju programskega inženirstva. Vsak element pripada izključno enemu področju;
- Postavka kontrolnega seznama (Check list item): za vsako postavko kontrolnega seznama je potrebno verificirati stanje;
- Kompetenca (Competency): kompetenca posameznika obsega sposobnosti, zmogljivosti, dosežke, znanja in veščine, ki morajo biti dosežene, da je določeno opravilo uspešno zaključeno. Kompetenca je tipično definirana stopenjsko od 0 (zmore asistirati) do 5 (zmore inovirati);
- Omejitve (Constraints): omejitve, politike, pravila, ki jih mora tim upoštevati;
- Vpeljava (Enactment): dejanje, ki je povezano z vpeljavo metode za določen namen, tipično prizadevanje;
- Prizadevanje (Endeavor): aktivnost ali množica aktivnosti, ki so ciljno usmerjene;
- Stalnica (Invariant): stalnica je predlog pojava jezikovnega elementa. Ima logično vrednost »res«, če je pojav uporabljen v jezikovnem konstruktu skladno z namero specifikacije. Stalnica je v konceptualnem smislu več kot spremenljivka. V splošnem gre za lastnost, ki je vedno »res«;
- Jedro (Kernel): jedro sestavlja množica elementov, ki predstavljajo osnovo za opis prizadevanj programskega inženiringa;

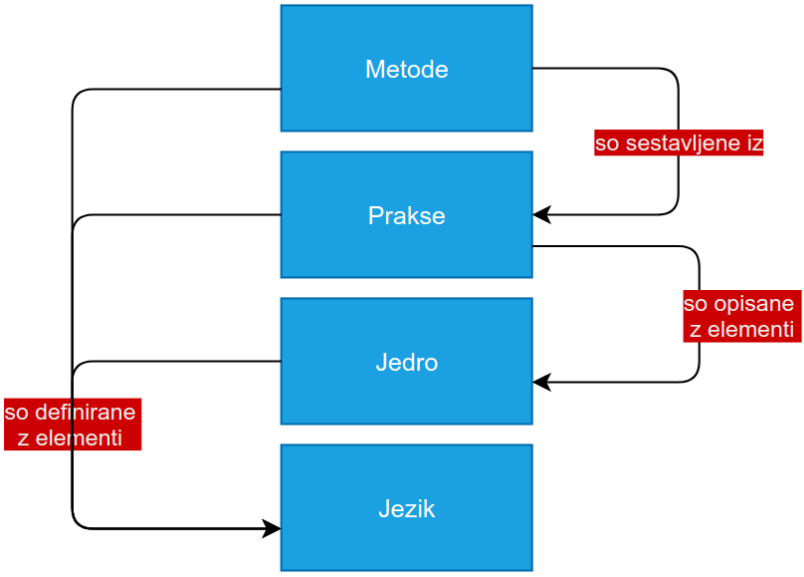
- Metoda (Method): metoda je sestavljena iz jedrnih elementov in množice praks, ki izpolnjujejo določen namen. Metoda je sinonim za način dela tima in mu zagotavlja pomoč in vodilo pri izvedbi nalog. Potek napora uporablja pojav metode. Ta pojav vsebuje pojave alf, delovnih izdelkov, aktivnosti in podobnih stvari, ki jih v realnosti povzroči vloženi razvojni napor. Uporabljeni pojav metode vsebuje sklic na definiran pojav metode, ki sledi. Opomba avtorjev: v opisu termina »metoda« so implicitno uporabljeni objektno orientirani koncepti. Metoda je opredeljena kot razred, uporaba je opredeljena kot objekt (pojav), razred ima članske spremenljivke in metode (alfe, delovne izdelke, aktivnosti ipd.). Med članskimi spremenljivkami je tudi taka, ki kaže na naslednjo aktivnost;
- Priložnost (Opportunity): priložnost je množica okoliščin, ki omogočajo razvoj ali spremembo programskega sistema;
- Vzorec (Pattern): vzorec je opis strukture v praksi;
- Praksa (Practice): praksa je ponovljiv pristop za realizacijo določenega cilja;
- Zahtevane lastnosti (Requirements): zahtevane lastnosti določajo, kaj mora rešitev nasloviti, da bo izkoristila priložnost in zadovoljila deležnike;
- Vloga (Role): množica odgovornosti. Opomba avtorjev: vloga je tipično seznam pravic uporabnika, ki se nanašajo na uporabo programskih funkcij za manipulacijo s podatki. Predpostavljava, da so želeli kreatorji jezika in jedra Essence ločiti dejstvo, da ima uporabnik specificirane pravice od procesa dodeljevanja pravic, torej od dodeljevanja odgovornosti;
- Programski sistem (Software system): programski sistem sestavljajo programska oprema, strojna oprema in podatki. Izvajanje programske opreme realizira korist za uporabnika;
- Deležniki (Stakeholders): posamezniki, skupine, organizacije, ki na programski sistem vplivajo ali pa le-ta vpliva na njihovo delo;
- Stanje (State): stanje izraža situacijo, v kateri so izpolnjeni določeni pogoji;
- Graf stanj (State Graph): graf stanj je usmerjen graf z definiranimi pogoji prehoda stanj. Ima eno začetno stanje in eno ali več končnih stanj;
- Tim (Team): skupina posameznikov, ki je aktivno vključena v razvoj, vzdrževanje, dostavo ali podporo določene programske opreme;
- Prehod (Transition): prehod je usmerjena povezava v grafu stanj;
- Način dela (Way-of-working): prilagojena (prikrojena) množica praks in orodij, ki jih tim uporablja za izvedbo dela;

- Delo (Work): delo je definirano kot mentalne in fizične aktivnosti, ki jih opravi tim za izdelavo programskega sistema;
- Postavka dela (Work item): postavka dela je del dela, ki mora biti opravljen za zaključek dela. Ima konkreten rezultat in vodi v spremembo stanja ali potrditev trenutnega stanja. Postavka dela je lahko povezana z aktivnostjo, ni pa to nujno.

Jedro in jezik Essence je namenjen tako praktikom kot tudi ustvarjalcem novih metod razvoja programske opreme. Glavne značilnosti jedra in jezika Essence so:

- ločeno je »kaj« od »kako«. Jedro jezika omogoča artikulacijo "kaj", prakse in metode pa določajo »kako«;
- neodvisnost od velikosti projekta (veliki, srednji, mali), preprosta razširljivost;
- praktična podpora razvojnim prizadevanjem;
- osredotočenost na uporabo metode in ne na opis metode. Alfe v vsakem trenutku omogočajo izmeriti »zdravje« projekta;
- hitra gradnja novih metod z uporabo preskušenih praks v soglasju s timom razvijalcev;
- hiter, poceni, postopen začetek oz. vpeljava (npr. s karticami, jedrom ali obstoječo prakso);
- prilagodljivost za različne akterje v timu razvijalcev;
- podpora agilnosti, prilagajanje med potekom razvoja, prilagodljivost na obseg med razvojem, uporaba principa »ločevanje pristojnosti« (separation of concerns).

Arhitektura modela Essence obsega metode, prakse, jedro in jezik. Metode so sestavljene iz praks, prakse opisujejo jedrni elementi, metode, prakse in jedro pa so definirani z elementi jezika (slika 8).



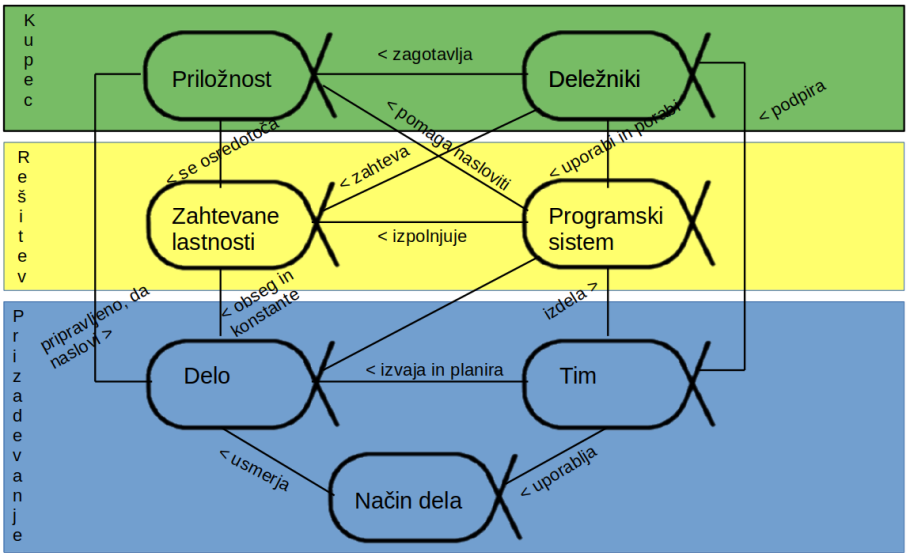
Slika 8: Arhitektura modela Essence obsega metode, prakse, jedro in jezik
Vir: lasten.

Jedro Essence se osredotoča na bistvene stvari in ni odvisno od metode za dokumentiranje rezultatov. Jezik je domensko specifičen za prakse in metode programskega inženirstva. V jeziku Essence je pomembnejša intuitivna grafična sintaksa kot formalna semantika. To ne pomeni, da semantika ni pomembna ali ni potrebna. Bistveno je, da mora biti opis na voljo v lahko razumljivem jeziku skupnosti razvijalcev.

Jedro jezika je organizirano na tri področja pozornosti:

- kupec: vse, kar je povezano z uporabo in izkoriščanjem programskega sistema. Na tem področju sta definirani dve alfi: priložnost in deležniki;
- rešitev: vse, kar je povezano s specifikacijo in razvojem programskega sistema. Na tem področju sta definirani dve alfi: zahtevane lastnosti in programski sistem;
- prizadevanje: vse, kar je povezano s timom in delom tega tima. Na tem področju so definirane tri alfe: delo, tim in način dela.

Na sliki 9 so prikazane povezave med alfami na posameznih področjih pozornosti.



Slika 9: Povezave med alfami na posameznih področjih pozornosti
Vir: lasten.

Definicija alf – s čim delamo?

- Priložnost: niz okoliščin, zaradi katerih je primeren razvoj ali sprememba programskega sistema. Priložnost artikulira razlog za nastanek novega ali spremenjenega programskega sistema. Predstavlja skupno razumevanje potreb deležnikov (interesnih skupin) in pomaga oblikovati zahtevane lastnosti za nov programski sistem z utemeljitvijo za njegov razvoj.
- Deležniki: zainteresirane strani; posamezniki, skupine ali organizacije, ki vplivajo na programski sistem ali le-ta vpliva nanje. Deležniki zagotavljajo priložnost in so vir zahtevanih lastnosti ter financiranja programskega sistema. Tudi člani tima so deležniki. Sodelovanje deležnikov je način podpiranja tima, da ta izdela sprejemljiv programski sistem.
- Zahtevane lastnosti: te določajo, kaj mora programski sistem nasloviti v zvezi s priložnostjo in tako zadovoljiti pričakovanja deležnikov. Pomembno je odkriti, kaj mora obsegati programski sistem, ter to sporočiti deležnikom. Zahtevane lastnosti so pomembne ves čas razvoja in testiranja programskega sistema.

- Programski sistem: sestavljajo ga programska oprema, strojna oprema in podatki. Programski sistem zagotavlja korist le, če deluje. To je glavni produkt programskega inženirstva in je lahko del večjega sistema, strojne opreme ali poslovna rešitev.
- Delo: dejavnost, ki vključuje duševni ali fizični napor, da bi dosegli rezultat. V kontekstu programskega inženiringa je delo vse, kar tim naredi za doseganje ciljev. Ti so odvisni od zahtevanih lastnosti, ki so skladne s priložnostjo. Slednje zagotovijo deležniki. Delo vodijo prakse, ki določajo način dela.
- Tim: skupina, ki se aktivno ukvarja z razvojem, vzdrževanjem, dostavo ali podporo določenega programskega sistema. Ena ali več skupin načrtuje in izvaja delo: ustvarja, posodablja in spreminja programski sistem.
- Način dela: prilagojen (prikrojen) nabor praks in orodij, ki jih tim uporablja za usmerjanje in podporo pri svojem delu. Tim razvija svoj način dela skupaj z razumevanjem svojega poslanstva in okolja. Pri svojem delu člani tima nenehno razmišljajo o svojem načinu dela in ga po potrebi prilagajajo trenutnemu kontekstu.

Definicija prostorov aktivnosti: kaj delamo?

Jedro vsebuje tudi nabor aktivnosti kot komplement alfam. Prostori aktivnosti bazirajo na aktivnostih v programskem inženirstvu. Na področju pozornosti »kupec« so aktivnosti:

- raziskovanje možnosti: to predstavlja ustvarjanje novega ali izboljšanega programskega sistema. Vključuje analizo priložnosti in identifikacijo deležnikov;
- razumevanje potreb deležnikov: sodelovanje z deležniki je nujno za razumevanje njihovih potreb in zagotavljanja pričakovanega rezultata. Identificiranje deležnikov in sodelovanje z njihovimi predstavniki omogoča boljšo opredelitev priložnosti;
- zagotavljanje zadovoljstva deležnikov: za pridobitev potrditve ustreznosti izdelanega programskega sistema je skupaj z deležniki nujno verificirati, da je bila priložnost prav naslovljena;
- uporaba sistema: opazovanje sistema v uporabi in ugotavljanje kako sistem koristi deležnikom.

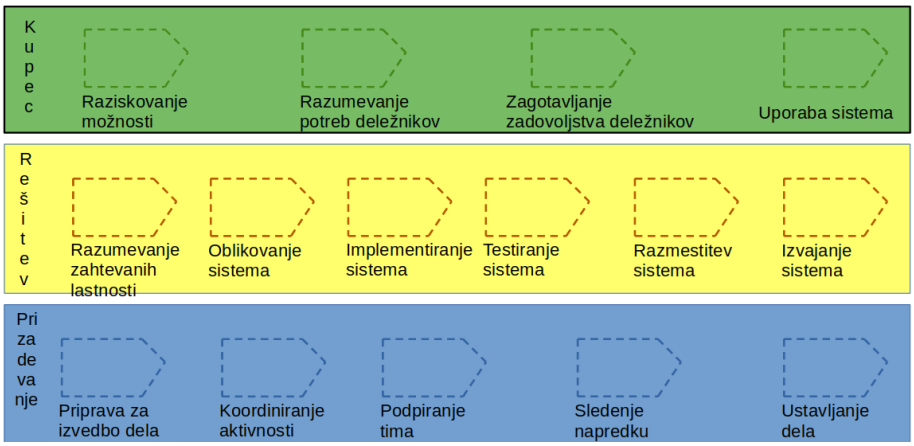
Na področju »rešitev« so aktivnosti:

- razumevanje zahtevanih lastnosti: vzpostavljanje skupnega razumevanja, kaj mora omogočati sistem;
- oblikovanje sistema: sistem je treba oblikovati tako, da ga je možno enostavno razvijati, spreminjati in vzdrževati. Upoštevati je treba tudi pričakovane prihodnje potrebe. Izdelati je treba zasnovo in arhitekturo sistema;
- implementacija sistema: izgradnja lahko vključuje tudi integracijo že izdelanih delnih rešitev. V tem delu se izvaja tudi odpravljanje slabosti in napak;
- testiranje sistema: preverjanje, ali izdelani programski sistem izpolnjuje definirane zahteve (zahtevane lastnosti);
- razmestitev sistema: s to aktivnostjo omogočimo uporabo posameznikom in skupinam izven ožjega razvojnega tima;
- izvajanje sistema: delovanje v živo.

Na področju »prizadevanja« so aktivnosti:

- priprava za izvedbo dela: vzpostavitev tima in delovnega okolja. Člani tima morajo razumeti nalogo in biti odločeni, da jo izvedejo;
- koordiniranje aktivnosti: koordiniranje in usmerjanje dela v timu. To vključuje tekoče načrtovanje, popravljanje načrtov dela in popolnjevanje tima;
- podpiranje tima: nudenje pomoči znotraj tima, promocija sodelovanja in izboljševanja načina dela;
- sledenje napredku: merjenje in ocenjevanje napredka;
- ustavljanje dela: ustavljanje dela v sklopu programskega inženirstva ter sprostitev članov s projekta razvoja.

Na sliki 10 so prikazani prostori aktivnosti po področjih pozornosti.



Slika 10: Prostori aktivnosti po področjih pozornosti

Vir: lasten.

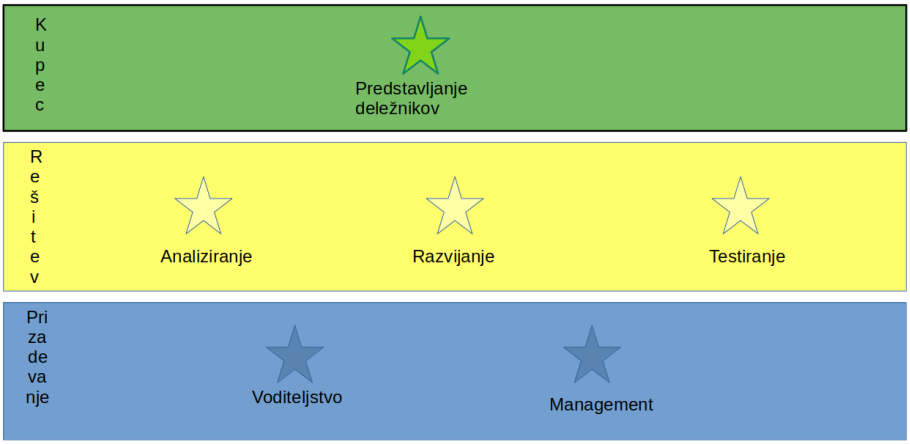
Definicija kompetenc: katere sposobnosti potrebujemo?

Jedro obsega tudi nabor kompetenc, ki so komplement alfam in prostorom aktivnosti. Glavne kompetence po področjih pozornosti so:

- kupec:
  - predstavljanje deležnikov. Ta kompetenca obsega sposobnost zbiranja, komuniciranja in uravnoteženja potreb deležnikov ter natančno predstavljanje njihovih pogledov;
- rešitev:
  - analiziranje: kompetenca obsega sposobnost razumeti priložnosti in potrebe deležnikov ter le-te transformirati v dogovorjeno in konsistentno množico zahtevanih lastnosti,
  - razvijanje: kompetenca obsega sposobnost oblikovanja in programiranja učinkovitega programskega sistema, ki je skladen z zahtevanimi standardi in normami v aktivnostih tima,
  - testiranje: kompetenca obsega sposobnost testiranja programskega sistema in verificiranja, da ustreza predpisanim zahtevanim lastnostim;
- prizadevanje:
  - voditeljstvo: ta kompetenca omogoča navdihovanje in motiviranje tima za uspešen zaključek dela ter doseganje cilja,

– management: kompetenca obsega sposobnost koordiniranja, planiranj in sledenja dela v timu.

Grafično so kompetence predstavljene kot zvezdice na področjih pozornosti (slika 11).



Slika 11: Grafično predstavljene kompetence na področjih pozornosti
Vir: lasten.

Kompetence imajo 5 splošnih stopenj:

– 1. stopnja - **asistiranje**: posameznik na tej stopnji dokazuje razumevanje ključnih konceptov in sledi navodilom. Obnaša se profesionalno. Sposoben je odgovoriti na osnovna vprašanja v svoji domeni. Sposoben je izvajati večino funkcij v svoji domeni. Zna slediti navodilom in opraviti osnovna opravila;
– 2. stopnja - **uporaba**: posameznik na tej stopnji je sposoben uporabiti koncepte v preprostem kontekstu in v rutinskih nalogah. Sposoben je sodelovati v timu. Zmore samozavestno izvesti rutinske naloge in enostavna opravila. Potrebuje pomoč pri obvladovanju zapletov in težav. Sposoben je sklepati o kontekstu in sprejemati racionalne odločitve;
– 3. stopnja - **mojster**: posameznik na tej stopnji je sposoben izvesti kontekst velike večine situacij. Ne potrebuje nadzora. Z lahkoto obvlada terminologijo domene. Zna komunicirati in razložiti svoje delo. Poda ali

vrne konstruktiven odziv. Pozna svoje omejitve in ve, kdaj mora poiskati pomoč eksperta;

- 4. stopnja - **prilagajanje**: posameznik na tej stopnji je sposoben presoditi, kdaj in kako uporabiti kontekste v zapleteni situaciji. Je sposoben prenosa znanja in omogoča drugim uporabo konteksta. Zmore rešiti kompleksne delovne zahteve. Je sposoben komunikator tudi z ljudmi izven domene. Zna usmerjati in pomagati članom v isti domeni. Sposoben je prilagoditi svoje delo tako, da delo poteka dobro tako v isti domeni kot tudi izven nje;
- 5. stopnja - **inoviranje**: posameznik na tej stopnji je prepoznaven strokovnjak, ki zmore razvijati koncepte ter navdihovati. Ima dolgoletne izkušnje in sledi dosežkom v domeni. Vrstniki mu priznavajo njegovo delo v domeni. Podpira člane pri reševanju kompleksnih problemov. Zna se pravilno odločiti, kdaj je potrebna inovacija ali delna sprememba ali rutinska reakcija. Sposoben je razviti inovativne in učinkovite rešitve v svoji domeni.

Posameznik mora za prehod na višjo stopnjo izkazovati vse predhodne kompetence. Na slikah 12, 13 in 14 prikazujemo miselne vzorce za alfe, prostore aktivnosti in kompetence. Slika 12 je komplement sliki 9, slika 13 je komplement sliki 10 in slika 14 komplement sliki 11.

Alfe imajo definirana stanja in pogoje za prehode med stanji. V nadaljevanju bodo prikazana vsa možna stanja alf in pogoji za prehod. Za poenostavitev spremljanja stanj alf so zelo priročne kartice, na katerih so ta stanja zapisana. Spremljanje napredka spominja na pristop kan-ban, ki se je razvil v japonski šoli kakovosti v industriji. Koristi te preproste grafične ponazoritve so:

- tim vidi stanje projekta,
- tim razume, kje mora vložiti napor za napredek,
- sledenje napredku in izvajanje pomembnih aktivnosti je nazorno prikazano,
- izogibanje napakam ali preprečevanje pozabljanja pomembnih aktivnosti,
- pri agilnih pristopih je prikaz dobra osnova za določitev ciljev sprinta,
- določitev od prakse neodvisnih kontrolnih točk, mejnikov in ciklov.

V nadaljevanju so predstavljene značilnosti alf. Stanja in pogoji za prehode so podani v tabelah od 1 do 7 (deležnik, priložnost, zahtevana lastnost, programski sistem, delo, način dela in tim).

Tabela 1: Stanja deležnikov in pogoji za prehod v stanje (področje pozornosti Kupec)

Stanje (deležnik)	Pogoji za prehod v stanje.
Prepoznan	Deležnik je prepoznan (identificiran). Predstavniki deležnika so vključeni. Definirane so odgovornosti deležnika.
Zastopan	Odgovornosti so dogovorjene. Pooblaščenci deležnika so avtorizirani za zastopanje. Dogovorjen je pristop k sodelovanju. Način dela je podprt - deležnik ravna skladno s tem (ga spoštuje).
Vključen	Predstavniki deležnika asistirajo. Zagotavljajo pravočasne povratne informacije in odločitve. Spremembe so bile takoj posredovane.
V dogovoru	Dogovorjena so minimalna pričakovanja. Predstavnik je zadovoljen z vključenostjo. Predstavnikov vložek (napor, informacije ipd.) je koristen in cenjen. Vložek tima je koristen in cenjen. Prioritete so jasne in perspektive uravnotežene.
Zadovoljen z razmestitvijo	Zagotovljene so povratne informacije deležnika. Sistem je pripravljen za razmestitev.
Zadovoljen z uporabo	Na voljo so povratne informacije o uporabi sistema. Sistem izpolnjuje pričakovanja

Tabela 2: Stanja priložnosti in pogoji za prehod v stanje (področje pozornosti Kupec)

Stanje (priložnost)	Pogoji za prehod v stanje.
Identificirana	Ideja o priložnosti je identificirana. Vsaj eden od deležnikov je zainteresiran za realizacijo. Ostali deležniki so identificirani.
Rešitev je potrebna.	Rešitev je identificirana. Potrebe deležnikov so vzpostavljene. Problem in ključni vzroki so identificirani. Potreba po rešitvi je potrjena. Obstaja vsaj en predlog rešitve.
Vrednost je ugotovljena.	Vrednost priložnosti je kvantificirana. Posledice rešitve so razumljene. Vrednost sistema je razumljena. Kriterij za uspešen zaključek je jasen. Rezultati so jasni in kvantificirani.
Izvedljiva	Rešitev je grobo nakazana. Rešitev je možna z določenimi omejitvami. Tveganja so sprejemljiva in obvladljiva. Rešitev je donosna. Razlogi za razvoj so razumljeni. Dokončanje je izvedljivo.
Naslovljena	Priložnost je naslovljena. Rešitev je vredna razmestitve. Deležniki so zadovoljni.
Koristi so pridobljene.	Rešitev je koristna. Vračilo investicije (ROI) je sprejemljivo.

Tabela 3: Stanja zahtevane lastnosti in pogoji za prehod v stanje (področje pozornosti Rešitev)

Stanje (zahtevana lastnost)	Pogoji za prehod v stanje.
Zasnovana	Deležniki se strinjajo o potrebni rešitvi. Uporabniki so identificirani. Deležniki, ki financirajo projekt, so identificirani. Priložnost je jasno opredeljena.
Omejena	Deležniki, ki bodo razvijali rešitev, so identificirani. Namen sistema je dogovorjen. Kriteriji uspešnosti so jasni. Vsi razumejo rešitev. Oblika zahtevanih lastnosti je dogovorjena. Upravljanje zahtevanih lastnosti je vzpostavljeno. Shema prioritet je jasna. Omejitve so identificirane in upoštevane. Predpostavke so jasno opredeljene.
Koherentna	Zahtevane lastnosti so predstavljene vsem. Izvor zahtevanih lastnosti je jasen. Konflikti so naslovljeni. Glavne značilnosti so jasne. Razloženi so

	ključni scenariji uporabe. Prioritete so jasne. Vpliv je razumljen. Tim ve in se strinja, kaj je potrebno dostaviti.
Sprejemljiva	Sprejemljiva rešitev je opisana. Spremembe so nadzorovane. Vrednost, ki bo realizirana, je jasna. Jasno je, kako nasloviti priložnost. Zahtevana lastnost je testabilna.
Naslovljena	Zahtevana lastnost je dovolj naslovljena, da je sprejemljiva. Zahtevane lastnosti in programski sistem sta skladna. Vrednost je jasno izražena. Programski sistem je vreden zagona.
Izpolnjena	Deležniki sprejmejo zahtevano lastnost. Ni drugih zahtevanih lastnosti, ki zavirajo. Zahtevane lastnosti so v celoti izpolnjene.

Tabela 4: Stanja programskega sistema in pogoji za prehod v stanje (področje pozornosti Rešitev)

Stanje (programski sistem)	Pogoji za prehod v stanje.
Arhitektura izbrana	Kriteriji za arhitekturo so dogovorjeni. Strojna platforma dogovorjena. Tehnologije izbrane. Meje sistema so poznane. Odločitve o organizaciji sistema so podane. Odločitev o nakupu, izgradnji ali ponovni uporabi sprejeta. Dogovorjena so ključna tehnična tveganja.
Dokazljiv	Ključne arhitekturne značilnosti so dokazljive. Sistem preskušen, zmogljivosti izmerjene. Kritične strojne konfiguracije dokazljive. Kritični vmesniki dokazljivi. Integracija z okoljem dokazljiva. Arhitektura sprejeta kot pripravljena za namen (fit-for-purpose).
Uporaben	Sistem je mogoče upravljati. Funkcionalnost preskušena. Zmogljivosti sistema sprejemljive. Odpovedi sprejemljive. Sistem je v celoti dokumentiran. Vsebina verzije znana. Jasna je dodana vrednost.
Pripravljen	Uporabniška dokumentacija je na voljo. Sistem je sprejet kot pripravljen za namen. Deležniki hočejo sistem. Operativna podpora vzpostavljena.
Operativen	Sistem je razpoložljiv za uporabo. Sistem živi. Dogovorjeni nivo storitve se izvaja.
Ukinjen	Zamenjan ali ukinjen. Podpore ni več. Ni avtoriziranih uporabnikov. Popravki ustavljeni.

Tabela 5: Stanja tima in pogoji za prehod v stanje (področje pozornosti Prizadevanje)

Stanje (tim)	Pogoji za prehod v stanje.
Zasnovan	Misija definirana. Omejitve poznane in definirane. Mehanizmi spreminjanja vzpostavljeni. Sestava definirana. Odgovornosti izpostavljene. Zahtevani nivo odločenosti dosežen. Zahtevane kompetence so identificirane. Velikost tima določena. Pravila upravljanja določena. Model voditeljstva izbran.
Oblikovan	V timu je dovolj članov. Pravila so razumljena. Način dela je razumljen. Člani predstavljeni. Posameznikove odgovornosti potrjene in skladne s kompetencami. Člani sprejemajo delo. Zunanji sodelavci so identificirani. Mehanizem komunikacije definiran. Člani so zavezani timu.
Sodeluje.	Tim deluje enotno. Komunikacija odprta in jasna. Tim je osredotočen na misijo. Člani se med seboj poznajo.
Deluje.	Dosledno izpolnjevanje dogovorjenih obveznosti. Nenehno prilagajanje spremembam. Naslavlja težave. Ponovno delo in zaostanki so minimizirani. Nepotreben napor stalno odpravljajo.
Ukinjen	Odgovornosti izpolnjene. Člani so na voljo drugim timom. Misija zaključena.

Tabela 6: Stanja dela in pogoji za prehod v stanje (področje pozornosti Prizadevanje)

Stanje (delo)	Pogoji za prehod v stanje.
Inicializirano	Zahtevani rezultati so jasni. Omejitve so jasne. Deležnik, ki financira projekt, znan. Iniciator znan. Deležniki, ki bodo opravili delo, so znani. Vir financiranja je jasen. Prioritete so jasne.
Pripravljeno	Zavezanost je podana. Stroški in napori so ocenjeni. Razpoložljivost virov razumljena. Izpostavljenost tveganjem razumljena. Kriteriji sprejemljivosti so vzpostavljeni. Osnovna razdelitev dela znana do mere, ki omogoča začetek. Naloge so identificirane, pomembnost posameznih nalog je določena. Verodostojen načrt je vzpostavljen. Financiranje deluje. Vsaj en član tima je pripravljen. Točke integracije so definirane.
Začeto	Razvoj začet. Napredek spremljan. Definicija opravljenega obstaja. Naloge napredujejo.
Nadzorovano	Naloge končane. Neplanirano delo nadzorovano. Tveganja nadzorovana. Ocene prilagojene tako, da odražajo zmogljivost. Napredek merjen. Popravila oz. ponovno delo nadzorovano. Zaveze so konsistentno izpolnjene.
Zaključeno	Obstajajo le še administrativne naloge. Rezultati so doseženi. Razviti sistem je sprejet.
Zaprto	Lekcija je naučena. Metrike razpoložljive. Vse je arhivirano. Proračun usklajen in zaprt. Tim sproščen za naslednjo misijo. Ni nedokončanih nalog.

Tabela 7: Stanja načina dela in pogoji za prehod v stanje (področje pozornosti Prizadevanje)

Stanje (način dela)	Pogoji za prehod v stanje.
Načela vzpostavljena	Tim aktivno podpira načela (principe). Deležniki se strinjajo s principi. Potrebna orodja so dogovorjena. Pristop priporočen. Operativni kontekst razumljen. Omejitve prakse in orodij so poznane.
Osnove vzpostavljene	Ključne prakse in orodja izbrana. Prakse, potrebne za začetek dela, so dogovorjene. Prakse in orodja, o katerih ni pogajanj, so identificirana. Vrzeli med razpoložljivim in potrebnim načinom dela so razumljene. Vrzeli v zmogljivostih so razumljene. Integriran način dela je na voljo.
V uporabi	Prakse in orodja v uporabi. Redno so pregledani. Prilagojeni kontekstu. Tim jih podpira. Mehanizmi povratne informacije so vzpostavljeni. Prakse in orodja podpirajo sodelovanje.
Na mestu	Uporablja ga celoten tim. Na voljo vsem v timu. Pregledan in prilagojen za celoten tim.
Deluje dobro	Predvidljiv napredek realiziran. Prakse uporabljene spontano. Orodja spontano podpirajo način dela. Nenehno uglaševanje.
Ukinjen	Ni več v uporabi. Naučene lekcije (izkušnje) se delijo.

Prostori aktivnosti so povezani s stanji alf preko vhodno-izhodne relacije. Za izvedbo aktivnosti morajo biti vzpostavljeni vhodni pogoji (stanja alf), rezultat pa je lahko novo stanje alf. Tabele 8, 9 in 10 prikazujejo vhodno-izhodne relacije na področjih pozornosti Kupec, Rešitev in Prizadevanje.

Tabela 8: Vhodno-izhodne relacije na področju pozornosti Kupec, Rešitev in Prizadevanje

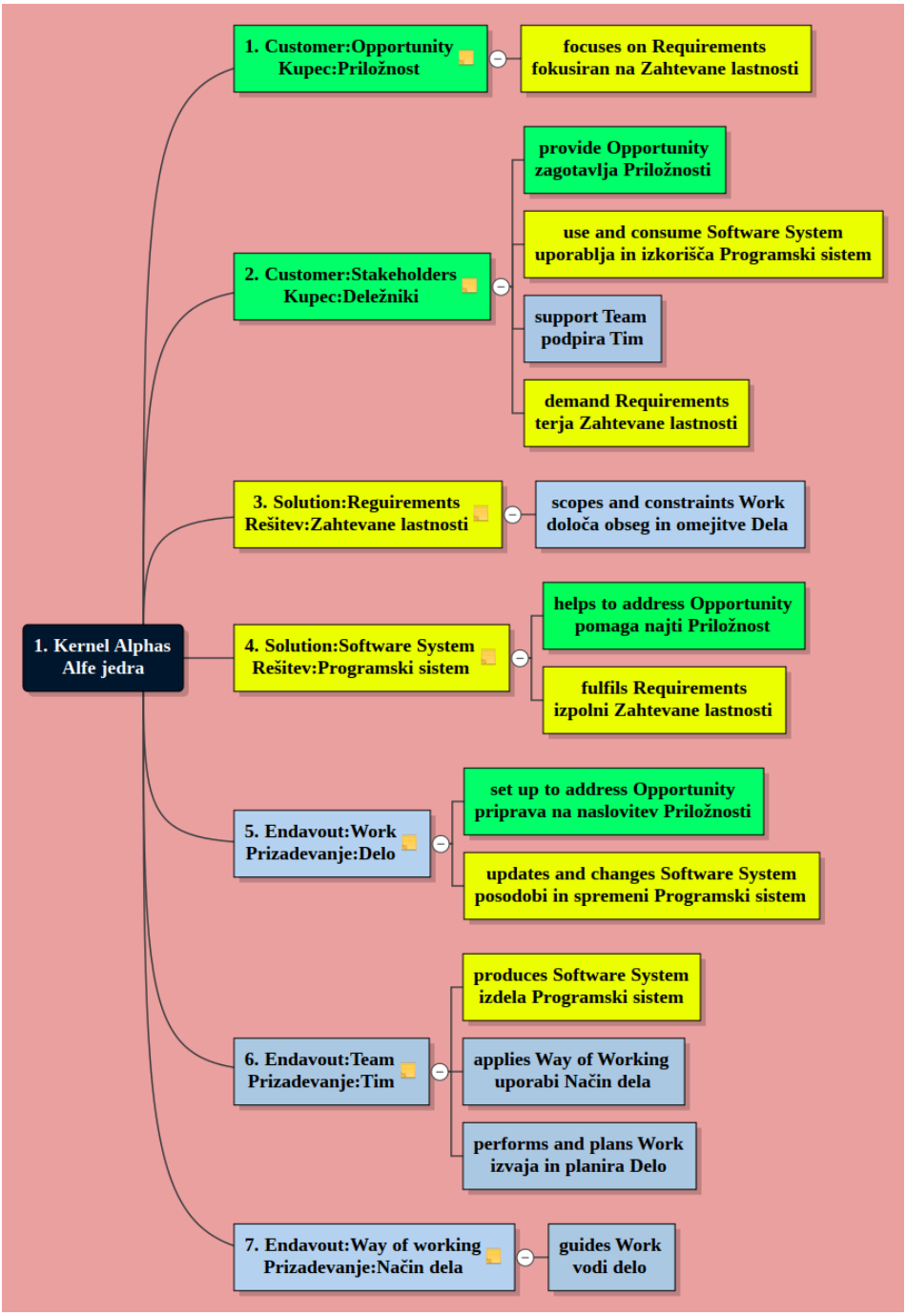
Prostor aktivnosti (področje Kupec)	Vhodne alfe	Vhodna stanja	Izhodna stanja
Raziskovanje možnosti: to predstavlja ustvarjanje novega ali izboljšanega programskega sistema. Vključuje analizo priložnosti in identifikacijo deležnikov.	-	-	Deležnik: prepoznan. Priložnost: identificirana.
Razumevanje potreb deležnikov: sodelovanje z deležniki je nujno za razumevanje njihovih potreb in zagotavljanja pričakovanega rezultata. Identificiranje deležnikov in sodelovanje z njihovimi predstavniki omogoča boljšo opredelitev priložnosti.	Zahtevana lastnost Programski sistem	Deležnik: prepoznan. Priložnost: vrednost je ugotovljena.	Deležnik: zastopan-vključen-v dogovoru. Priložnost: izvedljiva.
Zagotavljanje zadovoljstva deležnikov: za pridobitev potrditve ustreznosti izdelanega programskega sistema je skupaj z deležniki nujno verificirati, da je bila priložnost prav naslovljena.	Zahtevana lastnost Programski sistem	Deležnik: v dogovoru. Priložnost: vrednost je ugotovljena.	Deležnik: zadovoljen z razmestitvijo. Priložnost: naslovljena.
Uporaba sistema: opazovanje sistema v uporabi in ugotavljanje, kako sistem koristi deležnikom.	Zahtevana lastnost Programski sistem	Deležnik: zadovoljen z razmestitvijo. Priložnost: naslovljena.	Deležnik: zadovoljen z uporabo. Priložnost: koristi so pridobljene.

Tabela 9: Vhodno-izhodne relacije v področju pozornosti Rešitev

Prostor aktivnosti (področje Rešitev)	Vhodne alfe	Vhodna stanja	Izhodna stanja
Razumevanje zahtevanih lastnosti: vzpostavljanje skupnega razumevanja, kaj mora omogočati sistem.	Deležnik Priložnost Zahtevana lastnost Programski sistem Delo Način dela		Zahtevana lastnost: zasnovana-omejena-koherentna.
Oblikovanje sistema: sistem je potrebno oblikovati tako, da ga je možno enostavno razvijati, spreminjati in vzdrževati. Upoštevati je treba tudi pričakovane prihodnje potrebe. Izdelati je treba zasnovo in arhitekturo sistema.	Deležnik Priložnost Programski sistem Delo Način dela	Zahtevana lastnost: koherentna.	Zahtevana lastnost: sprejemljiva. Programski sistem: arhitektura izbrana.
Implementacija sistema: izgradnja lahko vključuje tudi integracijo že izdelanih delnih rešitev. V tem delu se izvaja tudi odpravljanje slabosti in napak.	Zahtevana lastnost Način dela	Programski sistem: arhitektura izbrana.	Programski sistem: dokazljiv-uporaben-pripravljen.
testiranje sistema: preverjanje, ali izdelani programski sistem izpolnjuje definirane zahteve (zahtevane lastnosti).	Način dela	Zahtevana lastnost: sprejemljiva. Programski sistem: arhitektura izbrana.	Zahtevana lastnost: naslovljena. Programski sistem: dokazljiv-uporaben-pripravljen.
Razmestitev sistema: s to aktivnostjo omogočimo uporabo posameznikom in skupinam izven ožjega razvojnega tima.	Deležnik Način dela	Programski sistem: pripravljen.	Programski sistem: operativen.
Izvajanje sistema: delovanje v živo.	Deležnik Priložnost Zahtevana lastnost Način dela	Programski sistem: pripravljen.	Programski sistem: ukinjen.

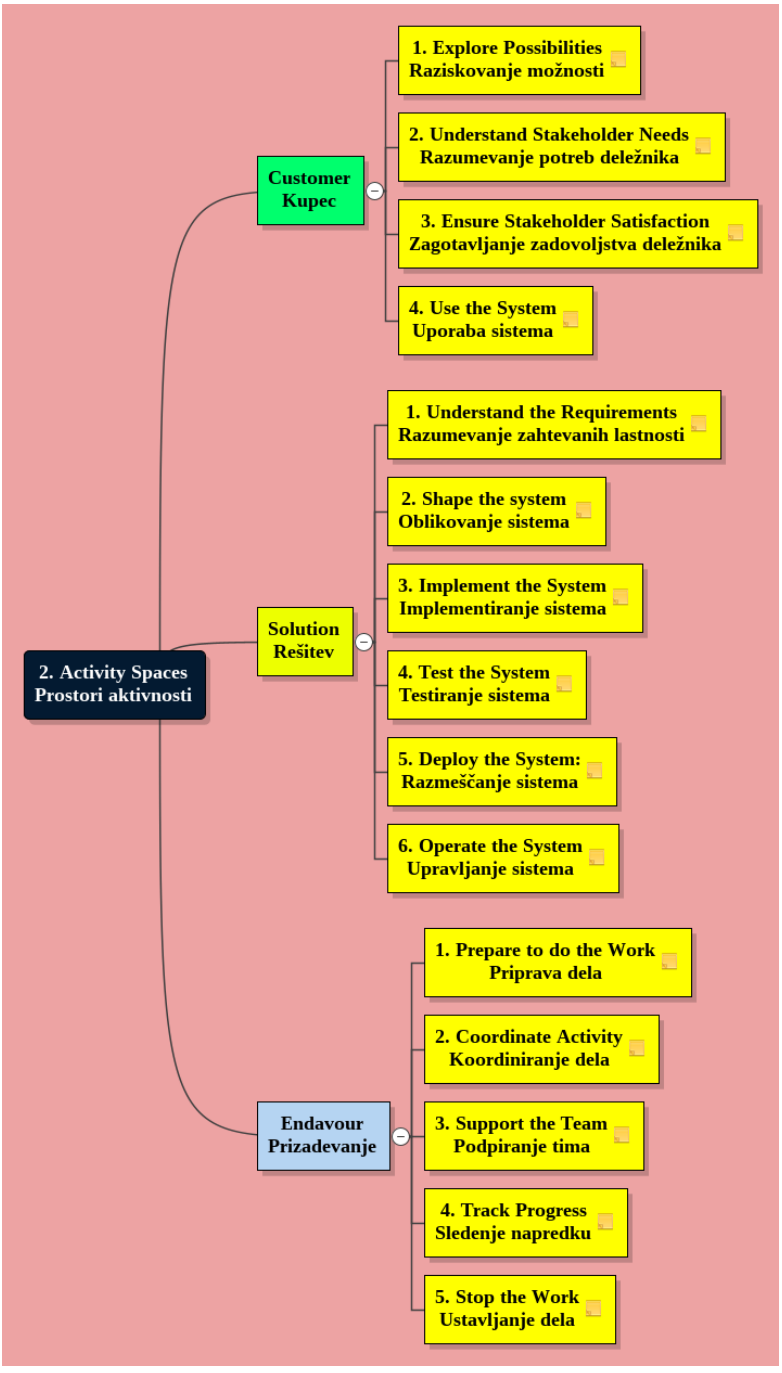
Tabela 10: Vhodno-izhodne relacije na področju pozornosti Prizadevanje.

Prostor aktivnosti (področje Prizadevanje)	Vhodne alfe	Vhodna stanja	Izhodna stanja
Priprava za izvedbo dela: vzpostavitev tima in delovnega okolja. Člani tima morajo razumeti nalogo in biti odločeni, da jo izvedejo.	Deležnik Priložnost Zahtevana lastnost	Deležnik Priložnost Zahtevana lastnost	Tim: zasnovan. Delo: inicializirano - pripravljeno. Način dela: načela vzpostavljena-osnove vzpostavljene.
Koordiniranje aktivnosti: koordiniranje in usmerjanje dela v timu. To vključuje tekoče načrtovanje, popravljanje načrtov dela in popolnjevanje tima.	Zahtevana lastnost Način dela	Tim: zasnovan. Delo: pripravljeno.	Tim: oblikovan. Delo: začeto-nadzorovano.
Podpiranje tima: nudenje pomoči znotraj tima, promocija sodelovanja in izboljševanja načina dela.	Delo	Tim: oblikovan. Način dela: osnove vzpostavljene.	Tim: sodeluje. Način dela: na mestu.
Sledenje napredku: merjenje in ocenjevanje napredka.	Zahtevana lastnost	Tim: sodeluje. Delo: začeto. Način dela: na mestu.	Tim: deluje. Delo: nadzorovano-zaključeno. Način dela: deluje dobro.
Ustavljanje dela: ustavljanje dela v sklopu programskega inženirstva ter sprostitev članov s projekta razvoja.	Zahtevana lastnost	Tim: deluje. Delo: zaključeno. Način dela: deluje dobro.	Tim: ukinjen. Delo: zaključeno. Način dela: ukinjen.



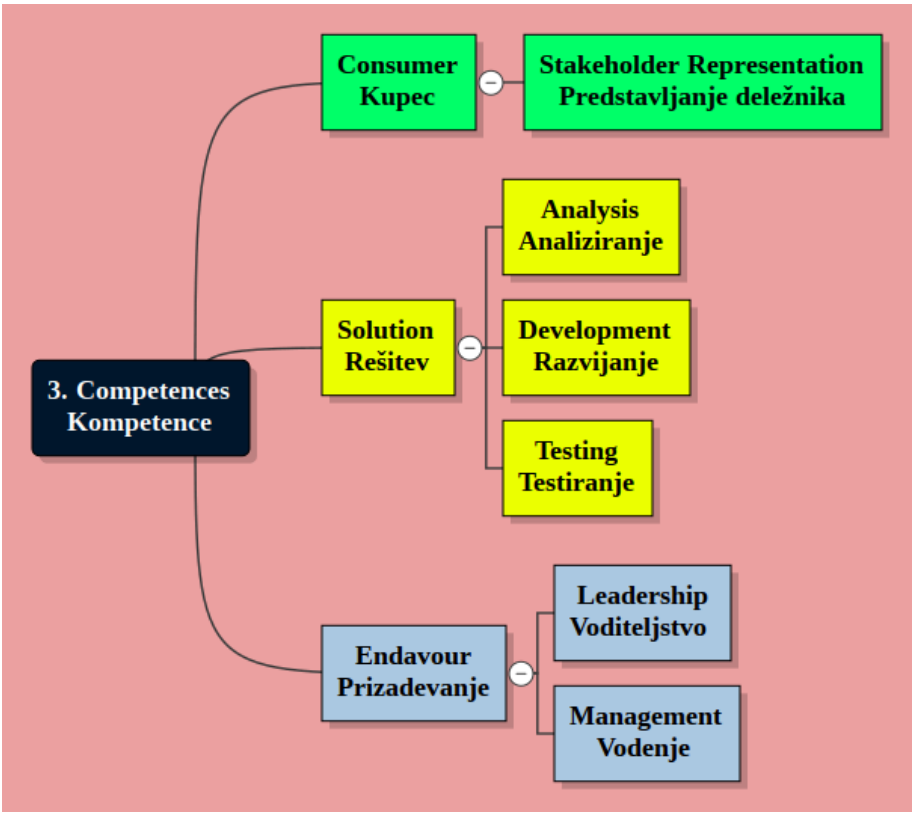
Slika 12: Miselni vzorec za alfe

Vir: lasten.



Slika 13: Miselni vzorec za prostore aktivnosti

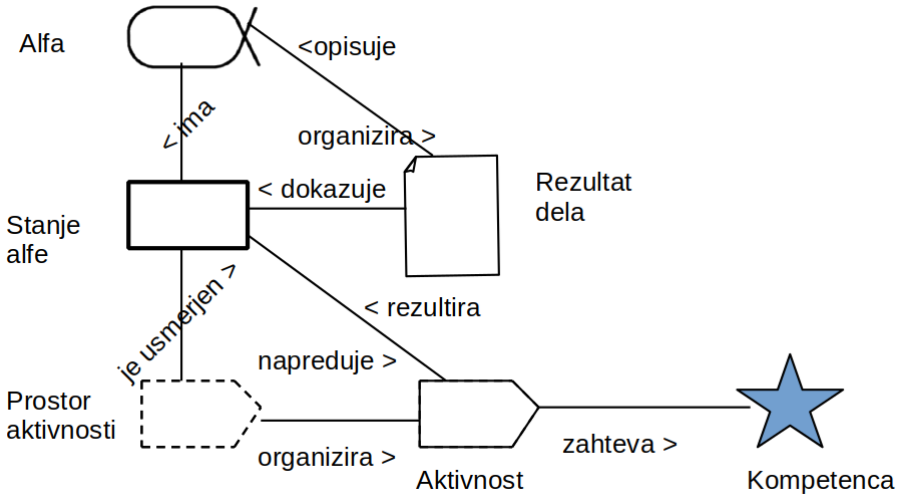
Vir: lasten.



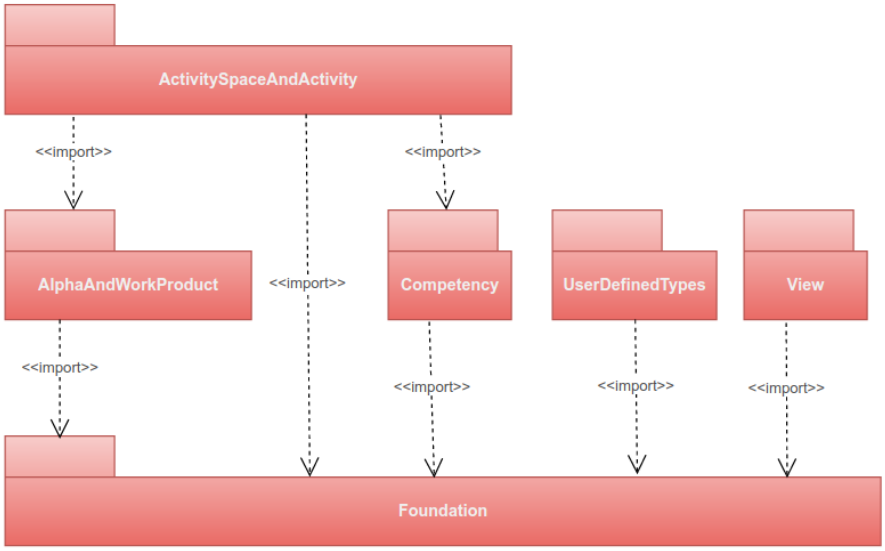
Slika 14: Miselni vzorec za kompetence
Vir: lasten.

3.2 Jezik

Specifikacija jezika je sestavljena s kombinacijo treh različnih tehnik: metamodela, formalnega jezika in naravnega jezika. Metamodel v poglavju 9.2 (Object Management Group, 2018) izraža abstraktno sintakso in nekatere omejitve strukturnih razmerij med elementi. Stalnice (angl. Invariant) zagotavljajo dobro oblikovana pravila statične semantike. Stalnice in nekatere druge operacije so realizirane z uporabo OC (Object Constraint Language). Sestava elementov kot tudi dinamična semantika sta opisani z uporabo naravnega jezika – angleščine. Na sliki 15 je prikazana neformalna predstavitev elementov in povezav med elementi jezika, na sliki 16 pa je prikazana struktura metamodela jezika Essence.



Slika 15: Neformalna predstavitev elementov in povezav med elementi jezika
Vir: lasten.



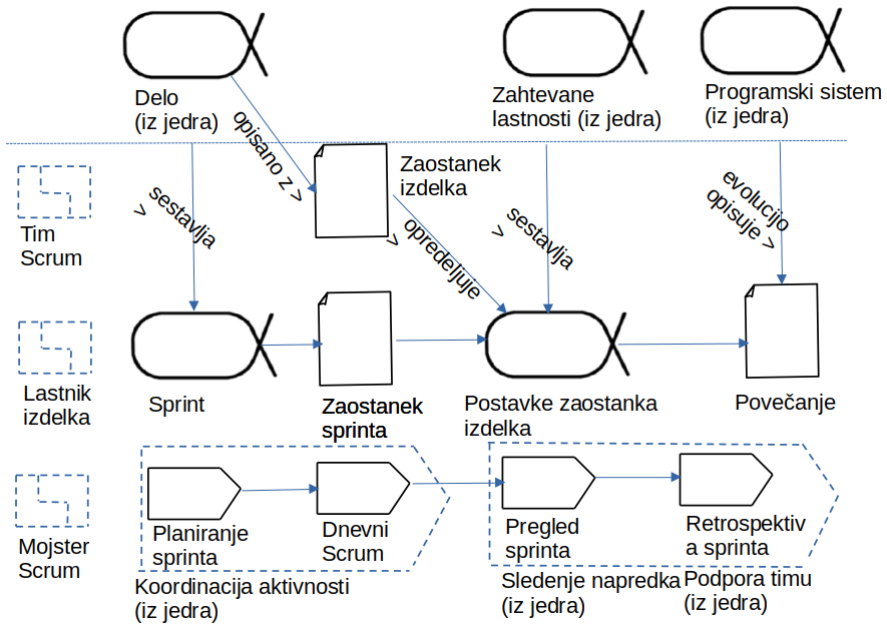
Slika 16: Struktura metamodela jezika Essence
Vir: lasten.

Namen paketov je:

- paket Foundation zagotavlja osnovne elemente {Checkpoint, ElementGroup, EndavourAssociacion, EndavourProperty, ExtensionElement, Kernel, LanguageElement, Library, MergeResolution, Method, Pattern, PatternAssociation, Practice, PracticeAsset, Resource, Tag}, vključno s super razredom;
- paket AlphaAndWorkProduct zagotavlja osnovne elemente {Alpha, AlpaAssociation, AlphaContainment, LevelOfDetail, State, WorkProduct, WorkProductManifest} za preproste oblike praks;
- paket ActivityAndActivityspace zagotavlja dodatne in naprednejše oblike praks {AbstractActivity, Action, ActionKind, Activity, ActivityAssociation, ActivitySpace, Approach, CompletionCriterion, Criterion, EntryCriterion};
- paket Competency zagotavlja dodajanje kompetenc praksam {Competency, CompetencyLevel};
- paket UserDefineďTypes omogoča dodajanje detajlnih informacij elementom v paketu Foundation {TypedPattern, TypedResource, TypeďTag, UserDefineďType};
- paket View zagotavlja uporabnikom interakcijo z relevantno podmnožico in relevantnimi detajli konstruktov jezika Essence {FeatureSelection, ViewSelection}.

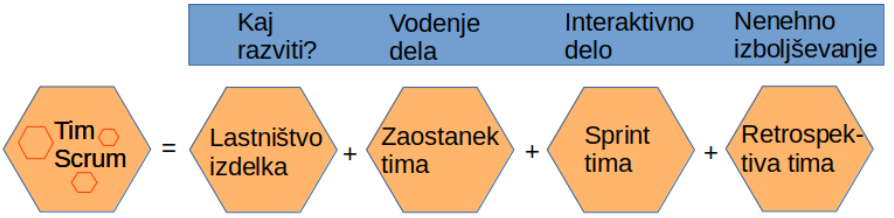
Jezik Essence omogoča sestavljanje in spreminjanje konstruktov. Vsak element jezika ima lasten nabor atributov ter možnost razširjanja (angl. extend) in sestavljanja (angl. merge). Celovita ocena prizadevanj programskega inženirstva je omogočena z uporabo specifične alfe in pripadajočega izdelka. Jezik definira tudi metamodel grafične sintakse za izmenjavo diagramov. Modeliramo lahko tudi kardinalnosti {npr. 0 ali mnogo, 0 ali 1, 1 ali več}. Besedna sintaksa je povzeta iz jezika SEMAT.

Modeliranje obstoječih praks Scrum je z jedrom in jezikom Essence dokaj preprosto. Primer: na sliki 17 je Scrum lite, na sliki 18 pa primer velikega projekta v Scrum. Oba primera sta povzeta po (Jacobson et al., 2019).



Slika 17: Metoda Scrum lite, prikazana v Essence

Vir: prirejeno po Jacobson et al., 2019.



Slika 18: Metoda Scrum za velik projekt v Essence

Vir: prirejeno po Jacobson et al., 2019.

Essence seveda ni omejen le na agilne metode. Z alfami in jezikovnimi konstrukti je možno opisati kakršenkoli proces razvoja – seveda tudi tradicionalnega in poljubno kompleksnega.

Zaključki

Dokaz kakovosti programske opreme je zadovoljstvo uporabnikov rešitve, ki svoje potrebe v delovnem področju v celoti, vedno in z minimalnim naporom zlahka izpolnijo. Njihovo zadovoljstvo pa mora sovpadati tudi s poslovno uspešnim projektom razvoja, ta pa je v veliki meri odvisen od lastnosti razvite rešitve. Merilo celovite uspešnosti projekta je poslovna korist tako naročnika kot tudi razvijalca programske rešitve. Predstavljeni model kakovosti programske opreme ISO/IEC 25010 sicer ne naslavlja poslovne koristi eksplicitno, jo pa vsebuje implicitno. Primer: obdobje vzdrževanja programske rešitve je običajno najdaljše in najdražje tako za naročnika kot tudi za razvijalca, z atributom vzdrževalnost pa model nakazuje tudi upoštevanje stroškovnega vidika. Analogija obstaja tudi pri drugih kriterijih kakovosti izdelka.

V iskanju metod programskega inženirstva pomeni Essence zagotovo pomemben korak v razvoju te discipline. Nivo abstrakcije na eni strani in možnost naslavljanja konkretnih praks, aktivnosti, procesov in okoliščin razvoja programske opreme omogočata Essence neslutene možnosti uporabe in nenazadnje izboljševanje statistike neuspešnih projektov v informacijski dejavnosti.

Povezava stabilnosti, ki jo predstavlja standard ISO/IEC 25010, in izjemna prilagodljivost jedra ter jezika Essence odpirata priložnost povezovanja in preseganja nedotakljivosti posamičnih področij. Vključitev kriterijev kakovosti v razvojni proces, ki ga izvajamo z metodo, bazirano na jedru in jeziku Essence, odpira vsaj dve pomembni polji v začetnih fazi razvoja rešitve:

- kriterije kakovosti izdelka, kot so funkcijska stabilnost, učinkovitost delovanja, združljivost, uporabnost, zanesljivost, varnost, vzdrževalnost, prenosljivost, se preslika v Zahtevane lastnosti v Essence;
- kriterije kakovost v uporabi, kot so uspešnost, učinkovitost, zadovoljstvo, odsotnost tveganj in obravnava konteksta, se preslika v Priložnosti v Essence.

V nadaljnjih korakih razvoja je tako vzpostavljena kontrolna zanka, ki omogoča minimiranje odklonov od kakovostne programske rešitve. Velik pomen takega pristopa je v neodvisnosti od metode razvoja, pa naj ta bazira na tradicionalnem življenjskem ciklu ali pa na agilnem pristopu.

Zahvala

Raziskava je bila podprta s strani Javne agencije za raziskovalno dejavnost Republike Slovenije v okviru programa P5-0018 – Sistemi za podporo odločanju v digitalnem poslovanju.

Literatura

Akbar, M. A., Sang, J., Khan, A. A., Fazal-E-Amin, Nasrullah, Shafiq, M., Hussain, S., Hu, H., Elahi, M., & Xiang, H. (2017). Improving the quality of software development process by introducing a new methodology-Az-model. IEEE Access, 6, 4811–4823. https://doi.org/10.1109/ACCESS.2017.2787981

Boehm, B. W., Brown, J. R., Kaspar, J. R., Lipow, M. L. & MacCleod, G. (1978). Characteristics of Software Quality. New York: American Elsevier.

Chaudhary, M., & Chopra, A. (2017). CMMI for Development. In CMMI for Development. https://doi.org/10.1007/978-1-4842-2529-5

Dendere, R., Janda, M., & Sullivan, C. (2021). Are we doing it right? We need to evaluate the current approaches for implementation of digital health systems. *Australian Health Review*, 778–781. https://doi.org/10.1071/AH20289

Dromey R. G. (1995). A model for software product quality. IEEE Trans. Softw. Eng., 21 (2) (1995), pp. 146-162, 10.1109/32.345830

ISO & IEC. (2001). ISO/IEC 9126-1:2001. Software engineering — Product quality — Part 1: Quality model. (https://www.iso.org/standard/22749.html)

ISO & IEC. (2017). ISO/IEC 25010 Software Quality Model. (https://www.iso.org/standard/35735.html)

ISO/IEC/IEEE. (2018). *ISO / IEC / IEEE 29148 Systems and software engineering - Life cycle processes - Requirements engineering.*

Jacobson, I., Lawson, H., Ng, P.-W., McMahon, P. E., & Goedicke, M. (2019). *The Essentials of Modern Software Engineering.*

McCall, J. A., Rihcards, P. K., Walters, G. F. Factors in Software Quality, Volumes I, II, and III. US Rome Air Development Center Reports, US Department of Commerce, USA, 1977.

Object Management Group. (2018). Kernel and Language for Software Engineering Methods (Essence). *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, *Versión 1.2*, 300. https://www.omg.org/spec/Essence/1.2

Park, J. S., Jang, J., & Lee, E. (2018). Theoretical and empirical studies on essence-based adaptive software engineering. *Information Technology and Management*, *19*(1), 37–49. https://doi.org/10.1007/s10799-016-0273-5

Reel J. S. (1999), Critical success factors in software projects, *IEEE Software*, vol. 16, no. 3, pp. 18-23, May-June 1999, doi: 10.1109/52.765782.