UNIVERZA V LJUBLJANI
EKONOMSKA FAKULTETA

MAGISTRSKO DELO

SODOBNE TEHNOLOGIJE ZA GRADNJO POSLOVNIH PROGRAMSKIH REŠITEV

Ljubljana, maj 2016 TEO VECCHIET

IZJAVA O AVTORSTVU

Spodaj podpisani Teo Vecchiet, študent Ekonomske fakultete Univerze v Ljubljani, izjavljam, da sem avtor zaključnega magistrskega dela z naslovom Sodobne tehnologije za gradnjo poslovnih programskih rešitev, pripravljenega v sodelovanju s svetovalcem prof. dr. Tomaž Turkom.

Izrecno izjavljam, da v skladu z določili Zakona o avtorski in sorodnih pravicah (Ur. l. RS, št. 21/1995 s spremembami) dovolim objavo magistrskega dela na fakultetnih spletnih straneh.

S svojim podpisom zagotavljam, da

- je predloženo besedilo rezultat izključno mojega lastnega raziskovalnega dela;
- je predloženo besedilo jezikovno korektno in tehnično pripravljeno v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, kar pomeni, da sem
    - poskrbel(-a), da so dela in mnenja drugih avtorjev oziroma avtoric, ki jih uporabljam v zaključnem magistrskem delu, citirana oziroma navedena v skladu z Navodili za izdelavo zaključnih nalog Ekonomske fakultete Univerze v Ljubljani, in
    - pridobil(-a) vsa dovoljenja za uporabo avtorskih del, ki so v celoti (v pisni ali grafični obliki) uporabljena v tekstu, in sem to v besedilu tudi jasno zapisal(-a);
- se zavedam, da je plagiatorstvo – predstavljanje tujih del (v pisni ali grafični obliki) kot mojih lastnih – kaznivo po Kazenskem zakoniku (Ur. l. RS, št. 55/2008 s spremembami);
- se zavedam posledic, ki bi jih na osnovi predloženega zaključnega magistrskega dela dokazano plagiatorstvo lahko predstavljalo za moj status na Ekonomski fakulteti Univerze v Ljubljani v skladu z relevantnim pravilnikom.

V Ljubljani, dne _____ Podpis avtorja(-ice):_____

KAZALO

KAZALO SLIK

KAZALO TABEL

UVOD

Temeljno vprašanje današnjih podjetij na področju informatizacije poslovanja je, kako lahko organizacija doseže, pridobi informacijski sistem, ki bi ji bil pisan na kožo, torej najprimernejši za način poslovanja in potrebe po informacijah ter avtomatizaciji delovnih procesov. Informacijski sistemi so v sodobnih časih temelj poslovanja, velika večina organizacij svojega poslanstva ne bi bila sposobna uresničevati brez informacijskega sistema oz. poslovnih programskih rešitev.

Ključne spremembe na tem področju so se začele ob prehodu v informacijsko dobo, saj je za doseganje uspešnosti organizacije ključnega pomena kakovostna obdelava informacij, v nasprotju s klasično obdelavo surovin. Razloge gre iskati v tem, da je uvajanje informatizacije v poslovanje podjetij sprožilo povečanje konkurence in bistveno skrajšalo čas, v katerem se mora podjetje odzivati na spremembe v poslovnem okolju, hkrati pa se je tudi povečal nabor poslovnih prijemov, ki jih podjetje lahko uporabi za delovanje na trgu (Gradišar, Jaklič, Damij & Baloh, 2005, str. 56). Informacijski sistemi so torej postali temelj poslovanja podjetij in drugih organizacij v današnjem času. Večina teh ne bi bila sposobna poslovati brez informacijskega sistema, enako velja tudi za neprofitne organizacije, pedagoške ustanove, državno upravo ipd. (Kurbel, 2008, str. 3).

V literaturi se pogosto pojavlja izraz »informacijski sistemi«, vendar je potrebno poudariti, da se ta pojem med različnimi avtorji razume drugače oz. uporabi namesto pojma »poslovna programska rešitev«. To v sami naravi lahko zveni kot nekakšen večji sistem, brez katerega izvajanje nalog in delovanje podjetja skoraj ni mogoče, vendar je potrebno poudariti, da poslovne programske rešitve niso nujno orjaški sistemi, informacijski sistemi, temveč so to lahko tudi mikro moduli, samostojne manjše programske rešitve, ki bistveno pripomorejo k učinkovitemu delovanju organizacije in dodajajo vrednost (Fowler, 2002, str. 21).

Carr (2003, str. 41–49) je v svojem kontroverznem delu “IT doesn’t matter” že pred več kot desetletjem trdil, da obstajata dve možnosti, kako lahko podjetje pride do poslovne programske rešitve, ki jo potrebuje za uspešno poslovanje. Lahko se odloči za razvoj svoje rešitve, ki bo zelo draga, lahko pa se odloči za standardizirano rešitev, ki je narejena po vzoru najboljših praks na nivoju vsakega poslovnega procesa, ki je prav tako standardiziran, in katere cena je bistveno nižja od lastnega razvoja. Izhajal je iz predpostavke, da so računalniški podatki, zapisi, v omenjenem primeru poslovne programske rešitve, enostavne za razmnoževanje, prav tako poslovni procesi, kar pomeni, da bo cena standardizirane rešitve zaradi ekonomije obsega bistveno nižja od cene lastnega razvoja. Omeniti velja trditev, ki jo v svoji knjigi omenja Kurbel (2008, str. 5), in sicer da je po njegovem mnenju smiselno, da so poslovne programske rešitve standardizirane le na nekaterih področjih, kot recimo pisarniške programske rešitve (primer Microsoft Office), tistih pa, ki izvajajo oz. podpirajo tako temeljne kot podporne poslovne procese podjetja, ni mogoče do te mere standardizirati, da bi jih organizacije lahko kupile ter uporabljale brez prilagajanja lastnim potrebam in zahtevam.

Poslovne programske rešitve običajno niso grajene v obliki samotnih otokov, temveč za optimalno delovanje potrebujejo povezavo, integracijo z drugimi že obstoječimi in bodočimi poslovnimi programskimi rešitvami. Organizacije namreč za delovanje uporabljajo večje število poslovnih programskih rešitev, ki vsaka posebej skrbi za delovanje in obdelovanje posameznih nalog. Tovrstne programske rešitve niso nujno grajene vse v enakem obdobju, nekatere so lahko starejše, nekatere bodo zgrajene v prihodnosti. To pomeni, da ni nujno, da bodo razvite z uporabo enakih tehnologij, vse pa morajo med seboj sodelovati, komunicirati in si deliti informacije in podatke.

Če tudi organizacija uspe vse zgoraj navedene težave oz. posebnosti poslovnih programskih rešitev obvladati, skoraj zagotovo naleti na težave na področju neskladij med poslovnimi procesi in poslovno programsko rešitvijo in podatki, ki so lahko tudi konceptualne narave. Kot primer Fowler (2002, str. 21) navaja razlike v razumevanju določenih pojmov med oddelki v organizaciji. Na primeru, kaj je za neko organizacijo kupec, bo eden izmed oddelkov mislil, da je to subjekt, s katerim trenutno poslujejo, drugi oddelek bo kot kupca dojemal subjekt, s katerim ima organizacija sklenjeno pogodbo, vendar že dalj časa ne poslujejo, tretji oddelek pa bo kot kupca obravnaval samo subjekt, ki kupuje izdelke in ne tistih, ki uporabljajo tudi poprodajne storitve. Rešitev opisanim težavam je konstantno upravljanje s podatki, ažuriranje in konsolidiranje ter usklajevanje s poslovnimi procesi in poslovno programsko opremo, zato Manouvrier in Menard (2008, str. 10–14) menita, da je za razvoj sodobne učinkovite poslovne programske rešitve nujno v proces razvoja vključiti tudi koncept managementa poslovnih procesov (v nadaljevanju MPP) (angl. *Business Process Management* – BPM), katerega naloga je pripraviti model poslovnega procesa, ki bo orkestriral izvajanje poslovnih programskih rešitev in pretakanje informacij.

Sicer se v literaturi zavedanje, da so poslovni procesi pomembni pri razvoju novih informacijskih sistemov, pojavlja že dlje časa. Österle na primer (1995, str. 17) trdi, da so poslovni procesi temelj za razumevanje kakršnega koli informacijskega sistema. Tudi eden večjih proizvajalcev poslovnih programskih rešitev in tehnologij za razvoj le-teh, podjetje Oracle, poudarja, da ni pravilo, da bi bile današnje poslovne programske rešitve naravnane samo k podatkom (angl. data-centric), temveč se vse pogosteje v praksi izkaže, da je primerneje poslovno programsko rešitev opisati kot večplastno implementacijo poslovnih procesov iz realnega sveta, ki predstavljajo skupek logično povezanih aktivnosti, prepletenih skozi več informacijskih sistemov, oddelkov in vlog, katere so lahko avtomatizirane ali ne (Jellema, 2011).

V svoji magistrski nalogi pa se ne bomo osredotočali na MPP kot celoto, temveč bo temeljni fokus magistrske naloge osredotočenje na fazo informatizacije poslovanja in poslovnih procesov ter še konkretneje na **jedrne razvojne tehnologije in razvojna orodja, ki se pri tem najmnožičneje uporabljajo**. Namen naloge je ugotoviti stanje na področju jedrnih razvojnih tehnologij in orodij za razvoj sodobne poslovne programske opreme, ki se uporablja za potrebe informatizacije poslovnih procesov.

Glede na zgoraj navedeno, torej koncept, da je poslovni proces osnovna podlaga za gradnjo informacijskih sistemov (poslovnih programskih rešitev), bomo v magistrski nalogi podrobneje raziskali to področje, torej tehnologije, ki omogočajo razvoj poslovnih programskih rešitev na podlagi informatizacije poslovnih procesov. Zanima nas, katere tehnološke rešitve so trenutno v uporabi, kako so se v času, v preteklosti, razvijale ter kam kažejo smernice v prihodnosti. V magistrskem delu bomo jedrne razvojne tehnologije, ki omogočajo razvoj poslovnih programskih rešitev in izhajajo in temeljijo na informatizaciji poslovnih procesov, poimenovali jedrne tehnologije.

Smith in Gray (2010, str. 2) sta jedrne tehnologije definirala kot osnovne gradnike, iz katerih so zgrajeni vsi tehnološki sistemi. Jedrne tehnologije so plod razvoja inženirjev, ki imajo cilj omogočiti gradnjo sistemov, ki bodo izpolnjevali človekove zahteve in želje. Pri gradnji informacijskih sistemov ključno vlogo odigrajo programske platforme (angl. *software platforms*) in razvojna ogrodja (angl. *development frameworks*). Vsako ogrodje nato omogoča različne koncepte razvoja poslovne programske rešitve z uporabo posameznih tehnoloških rešitev, programskih jezikov, standardov, vse to pa temelji na jedrnih tehnologijah, ki to omogočajo. Jedrna tehnologija določa, katero rešitev, pristop, se pri razvoju informacijskega sistema uporablja ter posledično tudi, katere so omejitve, ki jih je potrebno upoštevati. Uporabljena tehnologija določa način, na kakršen bo poslovna programska rešitev razvita, katera orodja za razvoj, namestitev in uporabo bodo na voljo (integrirana razvojna okolja, aplikacijski strežniki itd.) ter kako bodo lahko dodatne komponente dodane ali odstranjene.

V strokovni literaturi je zaslediti mnogo različnih tehnologij, večina pa jih ugotavlja, med drugim tudi podjetje Gartner (Sinur & Hill, 2010, str. 4-20) v študiji "Magic Quadrant for Business Process Management Suites", da je ob poplavi tehnoloških rešitev mogoče opaziti množično uporabo v praksi le nekaterih. S tehnološkim razvojem sta se namreč pojavili dve različni veji, za kateri se razvijalci lahko odločajo, ter se do neke mere med seboj izključujeta.

Prvo vejo predstavlja podjetje Microsoft s svojim ogrodjem .NET, ki za potrebe poslovnih programskih rešitev v kombinaciji s poslovnimi procesi ponuja celoten nabor lastnih jedrnih tehnologij, kot so Windows Workflow Foundation (v nadaljevanju WF), Windows Communication Foundation (v nadaljevanju WCF), Windows Presentation Foundation (v nadaljevanju WPF), hkrati pa ponuja tudi možnosti uporabe več programskih jezikov ter drugih jedrnih tehnologij, kot je npr. Business Process Execution Language (v nadaljevanju BPEL), ki naj bi predstavljal alternativno možnost tehnologiji WF. Samo podjetje pa razvijalskih paketov za razvoj obravnavanih poslovnih programskih rešitev ne trži, temveč je samo ponudnik jedrnih tehnologij.

Drugo vejo pa predstavljajo rešitve, ki temeljijo na javanskem ogrodju (različica Java EE), javanskih strojih (angl. *Java Machine*) ter v večini najmnožičneje uporabljajo tehnologije BPEL, Business Process Modeling Notation (v nadaljevanju BPMN), XML Process Definition Language (v nadaljevanju XPDL). Posebnost te veje je, da vsak izmed večjih proizvajalcev razvijalskih paketov osnovno javansko ogrodje prilagodi na svoj način, svojim potrebam, kar je

tudi glavni razlog, da se nato v večini primerov rešitve, ki temeljijo na javanskem ogrodju, obravnavajo ločeno ter so med seboj zelo različne. Primer so rešitve multinacionalnih podjetij Oracle, IBM ter SAP.

Temeljni raziskovalni vprašanji, na kateri nameravamo v nalogi odgovoriti sta, katere so najmnožičneje uporabljene sodobne jedrne tehnologije za gradnjo poslovnih programskih rešitev ter katere so ključne razlike med njimi in kakšne so bile smeri razvoja sodobnih jedrnih tehnologij za gradnjo poslovnih programskih rešitev ter kakšen bo njihov razvoj v prihodnje.

Raziskovalna naloga bo temeljila na kombinaciji tako deduktivnega kot induktivnega raziskovalnega pristopa. V prvem, drugem in tretjem poglavju bomo opredelili osnovne pojme, povzeli trditve in teoretične ugotovitve posameznih avtorjev, ki so povezane z obravnavano tematiko. Izhajali bomo iz strokovne literature ter virov z najnovejšimi teoretičnimi spoznanji s področja managementa poslovnih procesov, informatizacije poslovanja, razvojnih ogrodij in jedrnih tehnologij, ki so povezane z obravnavanimi razvojnimi ogrodji. Metodološki pristop, uporabljen v tem delu magistrske naloge, bo pregled strokovne literature domačih in tujih avtorjev.

V prvem delu četrtega poglavja bomo na podlagi pregleda obstoječih raziskav ugotovili katere so trenutno najmnožičneje uporabljene jedrne razvojne tehnologije. Opisali bomo njihove dosedanje ter bodoče strategije in smernice razvoja. Pri tem bomo izhajali iz informacij, pridobljenih na podlagi pregleda tehničnih publikacij in priročnikov avtorjev obravnavanih tehnologij, uradnih podpornih forumov, blogov in dnevnikov s katerimi upravlja proizvajalec dotične jedrne tehnologije. V drugem delu tega poglavja bomo nato predstavili in povzeli ugotovitve dosedanjih primerjav, raziskav in študij primerov obravnavanih jedrnih razvojnih tehnologij, ki so objavljene v strokovni literaturi. Ugotovitve bomo analizirali ter med seboj primerjali in nato predstavili rezultate primerjave.

Raziskovalni cilji, ki jih v nalogi zasledujemo so sledeči:

- prikazati prednosti in izzive uporabe MPP,
- izdelati pregled področja jedrnih razvojnih tehnologij in standardov, ki nastopajo na obravnavanem raziskovalnem področju,
- ugotoviti katere so temeljne jedrne tehnologije, ki nastopajo pri razvoju poslovne programske opreme na podlagi MPP,
- ugotoviti smeri razvoja posameznih tehnologij v času ter pregled in analiza trenutnega stanja na tem področju,
- poizkusiti predvideti, kam bodo smernice razvoja tehnologij na področju MPP kazale v prihodnje,
- na praktičnem primeru prikazati ključne razlike na med dvema najmnožičneje uporabljenima tehnologijama.

1 MANAGEMENT POSLOVNIH PROCESOV IN DELOVNI TOK

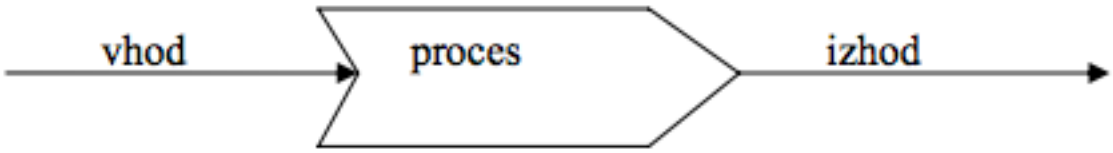
1.1 Management poslovnih procesov

1.1.1 Poslovni proces

Poslovni proces je, kot ga opredeljuje Khan (2004, str. 334) »skupek zaporednih ali vzporednih aktivnosti, ki jih izvajajo ljudje ali programske rešitve z namenom dosega skupnega cilja.« Hkrati pa Sharp in McDermott (2001, str. 196) poslovni proces opisujeta kot »popoln sklop aktivnosti, ki se odvijajo od začetka do konca ter skupaj prinašajo stranki korist«, natančneje pa je to »skupek med seboj povezanih aktivnosti, začetih kot odziv na nek dogodek, z izvajanjem pa za stranko procesa dosežejo nek cilj.«

Davenport (1993, str. 5) v svojem delu označi pojem procesa kot strukturiran, merljiv sklop aktivnosti, katerega cilj je ustvarjati proizvod ali storitev za kupca ali trg. Gre za posebno ureditev delovnih aktivnosti skozi čas in prostor, z začetkom in koncem ter z jasno opredeljenimi vhodi in izhodi. Ključen pri procesu je rezultat. Poslovni proces je torej skupek logično organiziranih zaporednih in/ali vzporednih aktivnosti, ki jih kot odziv na vhodni vložek izvajajo ljudje ali programske rešitve z namenom doseganja želenih rezultatov (izložkov) znotraj ene ali več organizacij. Osnovna oblika poslovnega procesa je prikazana na Sliki 1.

Slika 1: Shematski prikaz poslovnega procesa



Vir: A. Kovačič & V. Bosilj Vukšić, Management poslovnih procesov: prenova in informatizacija poslovanja s praktičnimi primeri, 2005, str. 29.

Če je proces pravilno načrtovan in definiran ter ima logično zaporedje stopenj ter vsi faktorji pozitivno vplivajo na posamezno stopnjo, potem bodo tudi rezultati procesa pozitivni. Če eden od faktorjev neustrezno vpliva na eno od stopenj procesa, potem rezultati procesa ne bodo popolnoma ustrezni (Marolt & Gomišček, 2005, str. 84).

1.1.2 Management poslovnih procesov

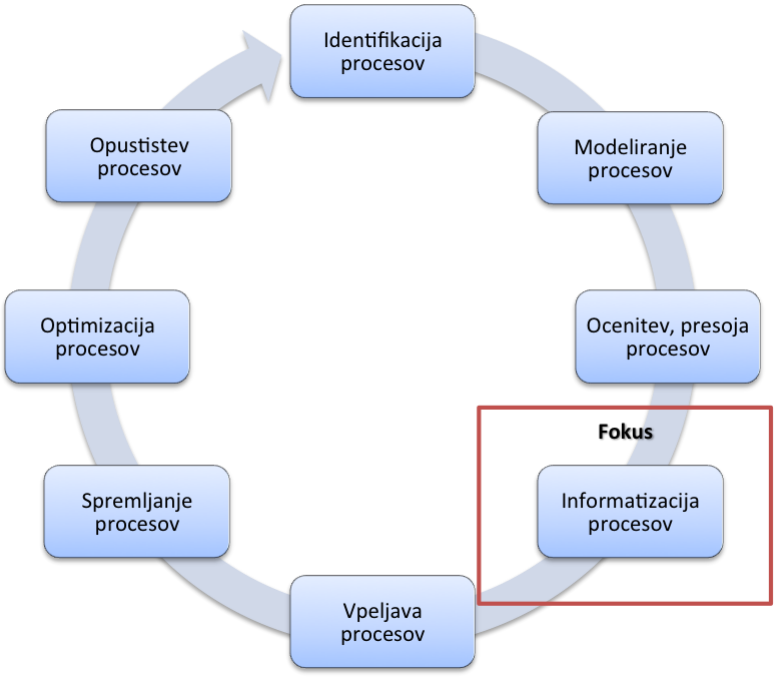
Ključna funkcija, ki se vse bolj uveljavlja v podjetjih, je načrtovanje, organiziranje, informatizacija in skrbništvo poslovnih procesov. Gre za funkcijo, ki na eni strani omogoča optimalen potek izvajanja procesnih aktivnosti, na drugi strani pa zagotavlja ustrezno informacijsko podporo izvajalcem teh aktivnosti. Takšno funkcijo v mnogih podjetjih poimenujejo kar management poslovnih procesov. Največkrat se kot organizacijska oblika

razvije iz službe za informatiko s ciljem obvladovanja poslovnih procesov podjetja oziroma premostitve znanega razkoraka med managementom in informatiko (Kovačič, Jaklič, Indihar Štemberger & Groznik., 2004, str. 40).

Management poslovnih procesov (angl. *Business Process Management*, v nadaljevanju MPP) je poslovni pristop k upravljanju sprememb poslovnih procesov. Predstavlja mnogo širše področje obravnave kot pri prenovi poslovnih procesov (angl. *Business Process Reengineering* – BPR). Usmerjen je v poslovno povezovanje procesov s procesi poslovnih partnerjev in z njihovimi informacijskimi sistemi. Znotraj podjetja je MPP usmerjen v razvoj platforme za integracijo poslovne strategije, poslovnega modela in poslovnih procesov podjetja z informacijskim modelom, arhitekturo in rešitvami, ki predstavljajo ključno infrastrukturo podjetja (Kovačič & Bosilj Vukšić, 2005, str. 15).

Cikel MPP obravnava celoten spekter sprememb poslovnega procesa, ki je sestavljen iz identifikacije, modeliranja, analize, informatizacije, vpeljave, spremljanja, optimizacije in opustitve procesov (Slika 2). Kljub temu pa cikel MPP ni omejen le na procese znotraj podjetja, obsega namreč tudi povezovanje organizacije s procesi in informacijskimi tehnologijami med poslovnimi partnerji (Kovačič et al., 2004, str. 70). Kot je z rdečim okvirjem označeno na Sliki 2, je v sklopu MPP fokus te magistrske naloge na informatizaciji poslovanja, saj je, kot menita Manouvrier in Menard (2008, str. 10–14), za razvoj sodobne učinkovite poslovne programske rešitve nujno v proces razvoja vključiti tudi koncept managementa poslovnih procesov.

Slika 2: Življenjski cikel managementa poslovnih procesov



Vir: T. Rozman, Upravljanje poslovnih procesov v podjetjih: principi, metode in človeški faktorji, 2010, str. 4.

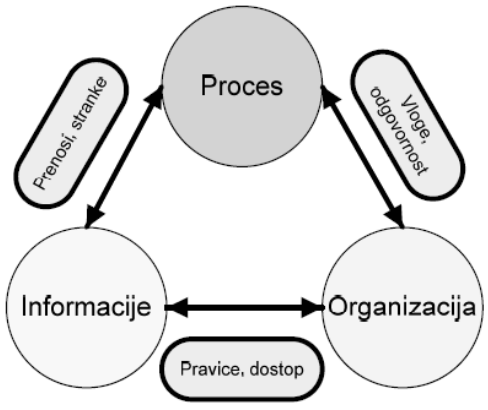
V literaturi se zavedanje, da so poslovni procesi pomembni pri razvoju novih informacijskih sistemov, pojavlja že dlje časa. Österle na primer (1995, str. 17) trdi, da so poslovni procesi temelj, za razumevanje kakršnega koli informacijskega sistema. Povezavo med poslovnimi procesi in informacijskim sistemom (poslovno programsko rešitvijo) je najlažje opisati na sledeči način: poslovni proces je skupek aktivnosti, informacijski sistem pa je, poleg drugih deležnikov, izvajalec in sodelavec teh aktivnosti (Lind, 2005).

Manouvrier in Menard (2008, str. 10–14) sicer MPP definirata kot proces priprave modela poslovnega procesa, katerega naloga je orkestracija izvajanja poslovnih programskih rešitev in pretakanja podatkov in informacij. Z modeliranjem je potrebno pripraviti modele poslovnih procesov. Na podlagi modela poslovnega procesa se razvije poslovna programska rešitev, ki se nato izvaja po navodilih dirigenta, tj. MPP-orodja oz. tehnologije.

1.1.3 Modeliranje in simulacija poslovnih procesov

Model je poenostavljena, abstraktna predstavitev realnega sveta, ki odraža predstavo ali nek pogled na stvarnost. Sestavljen je iz slike oziroma grafične predstavitve procesa, ki jo spremlja še opis značilnosti procesa, kot so vhodi, izhodi ter dogodki, ki sprožijo izvajanje procesa. Za samo izdelovanje modelov si pomagamo s tehnikami. Ta izraz označuje skupek običajno grafičnih oznak ali simbolov ter pravil, s katerimi izdelamo modele. Razlog za to, da modeliramo z grafičnimi tehnikami, je predvsem to, da so grafične predstavitve odlično sredstvo za razumevanje in ustvarjanje boljše predstave (Kovačič et al., 2004, str. 79).

Slika 3: Povezava med procesom, informacijami in organizacijo



Vir: L. Fischer, Workflow handbook, 2004, str. 299.

Pri modeliranju poslovnega procesa gre za nov ali obstoječ proces, ki je predstavljen kot model. Na tej stopnji so predlagane spremembe in izboljšave obstoječega procesa na podlagi analiz modela. Po drugi strani pa model predstavlja tudi podlago za informatizacijo poslovnega procesa. Najpomembnejše pri postavitvi modela poslovnega procesa je njegovo jasno razumevanje, kar pripomore k uspehu prenove poslovnih procesov in postavitvi sistema za

upravljanje s procesi. Pri modeliranju se predstavi tesna povezanost organizacije podjetja, poslovnega procesa in informacij, kot prikazuje Slika 3 (Fischer, 2004, str. 299).

Modeliranje služi lažjemu razumevanju poslovnega procesa, ker so ti lahko zelo kompleksni. Pomaga nam pri analizi celotne slike poslovanja in odkrivanju slabosti obstoječega modela poslovnega procesa. Po analizi pa nam modeli omogočajo ocenjevanje prenovljenega procesa in lažje razumevanje informacijskih potreb pri informatizaciji procesa (Kovačič et al., 2004, str. 9).

1.1.4 Informatizacija poslovnih procesov

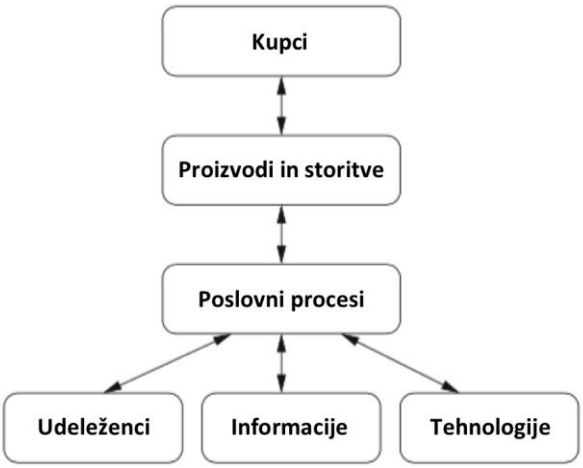
Management poslovnih procesov pomeni optimizacijo posredovanja dokumentov in informacij, ki nastajajo v procesu, njihovim izvajalcem, da le-ti nemoteno opravljajo delo in tako dosegajo poslovne cilje podjetja. Lahko je organizirano tradicionalno, delno ali popolnoma avtomatizirano. Tradicionalno zasnovan poslovni proces temelji na uporabi papirnih dokumentov. Sodoben poslovni proces je računalniško podprt, saj je le tako mogoča optimizacija in/ali avtomatizacija nalog (Kovačič & Bosilj Vukšić, 2005, str. 63). Avtomatizacija poslovnega procesa je torej prenova poslovnega procesa v smislu njegove informacijske podpore in iz tega sledeče optimizacije in/ali avtomatizacije nalog.

Osrednja značilnost sistemov za MPP je sposobnost informatizacije procesov. MPP torej združuje krmiljenje delovnih procesov in integracijo programskih rešitev. Drastično se zmanjša »mrtvi« čas (čas, ko se v procesu ne dogaja nič). Analitiki poslovnih procesov ocenjujejo, da je delež »mrtvega« časa pri tipičnih poslovnih procesih kar 90-odstoten (HandySoft Corporation, 2003, str. 11). Samo 10 odstotkov celotnega časa je uporabljenega za dejansko izvajanje procesa. Z informatizacijo procesa se »mrtvi« čas zelo zmanjša, kar je predvsem posledica tega, da naloge hitreje prihajajo do odgovornih oseb in da se v vsakem trenutku ve, kdo mora izvesti nalogo.

Z avtomatizacijo procesa je mogoče določena ročna opravila popolnoma avtomatizirati in jih prepustiti za to namenjeni programski rešitvi. Z uveljavljanjem spletnih storitev (angl. *Web Service*) se povezovanje poslovnih procesov z obstoječimi programskimi rešitvami v podjetju močno poenostavlja. Pri podprtju poslovnega procesa z uporabo orodja za MPP (angl. *Business Process Management System*, v nadaljevanju BPMS) imajo organizacije priložnost optimizirati poslovni proces in možnost vključevanja poslovnih partnerjev in kupcev. Z BPMS je moč doseči večjo transparentnost izvajanja procesa ter možnost prilagajanja svojih procesov ter prenos sprememb v uporabo v relativno kratkem času.

Cilj informatizacije poslovanja v sklopu MPP je čim manj uporabe zahtevnih tehnik programiranja z uporabo izključno programske kode, lažji razvoj programskih rešitev, narejenih »po meri« organizacije, prehod iz programiranja k »montaži« oz. »gradnji« (angl. *from programming to assembling*). Hkrati informatizacija omogoča učinkovito izrabo resursov organizacije, sama ločitev procesne logike od programske kode omogoča enostavnejšo prenovo poslovanja in nadgradnjo tako poslovanja kot poslovnih programskih programskih rešitev in informacijskih sistemov (Dumas, van der Aalst & ter Hofstede, 2005, str. 6).

Slika 4: Shema odvisnih povezav v MPP



Vir: M. Dumas et al., Process-Aware Information Systems, 2005, str. 6.

Na Sliki 4 je na nazoren način prikazana povezava med poslovnimi procesi in tehnologijami. Bistvo sheme je prikazati vzajemne učinke šestih gradnikov poslovnih programskih rešitev, ki izhajajo iz informatizacije poslovanja: Zadovoljstvo kupcev je odvisno od kvalitete proizvodov oz. storitev, katerih kvaliteta je odvisna od samih poslovnih procesov organizacije, kvaliteta izvajanja teh pa je odvisna od udeležencev v procesu, informacij in tehnologije. Prav tako pa so odvisna razmerja v obratni smeri (Dumas et al., 2005, str. 6).

1.2 Delovni tok

1.2.1 Kaj je delovni tok

Današnje organizacije se s konkurenco srečujejo na lokalni in globalni ravni, na vseh področjih delovanj, kar vodi do konstantne potrebe po izboljševanju produktivnosti. Težave, s katerimi se najpogosteje srečujejo organizacije in zaradi katerih posegajo po orodjih za delo z delovnimi tokovi, so sledeče (Ellis, 1999, str. 29):

- povečan obseg administrativnih nalog;
- zunanji pritiski za povečanje učinkovitosti;
- notranji pritiski za povečanje učinkovitosti;
- želja zaposlenih po boljšem plačilu in boljših delovnih razmerah.

Moderne organizacije običajno uporabljajo množico programskih rešitev, ki podpirajo njihove potrebe po informatizaciji poslovanja oz. so namenjene pomoči uporabniku pri informatizaciji. V to skupino spadajo programske rešitve za upravljanje z besedili, s preglednicami in podobne. Takšnih programskih rešitev je na tržišču veliko, manj je pa takšnih, ki so namenjene podpori pri sodelovanju različnih skupin ljudi. Eno najpopularnejših informacijskih orodij za upravljanje

sodelovanja med ljudmi in skupinami ljudmi je delovni tok (angl. *workflow*) (Ellis, 1999, str. 29). Predhodni odstavek se najbrž večini zdi nerazumljiv zaradi uporabe besede »delovni tok« za označevanje informacijskega orodja. Dejstvo pa je, da se v strokovni literaturi na področju informacijske tehnologije s tem izrazom označuje orodja za upravljanje z delovnimi tokovi. Gre za »sistem, ki pomaga organizacijam pri opredeljevanju, izvajanju, nadzoru in koordiniranju toka delovnih prigodkov (angl. *work cases*) znotraj pisarniškega okolja« (Ellis, 1999, str. 30).

Besedna zveza delovni tok v Slovarju slovenskega knjižnega jezika ne obstaja, je pa besedna zveza razložena na spletni strani terminološkega slovarja informatike "Islovar", kjer je delovni tok opredeljen kot "natančno opredeljeno in informatizirano zaporedje opravil v aktivnosti, poslovnem procesu ali njegovem delu". Na enak način delovni tok opredeljujeta tudi Delbecq in Van de Ven (1971, str. 466–492), kjer pojem tudi širše razložita kot koordiniran nabor aktivnosti, ki se izvajajo za doseg zastavljenega cilja. Iz tega sledi, da z upravljanjem delovnega toka (angl. *workflow management*) želimo doseči krmiljenje teh aktivnosti v organizaciji tako, da bo delo opravljeno na učinkovit način, ob pravem času in s strani prave osebe, z uporabo pravega orodja.

Organizacija Workflow Management Coalition (Workflow Management Coalition, 1996), konzorcij za opredelitev standardov na področju sistemov za management delovnih tokov, katerega ustanovni člani so tudi multinacionalna podjetja, kot sta IBM in Hewlett-Packard, pojem širše razlaga kot »avtomatizacijo poslovnih procesov, v celoti ali delno, pri kateri gre za posredovanje informacij in/ali dokumentov od enega uporabnika do drugega (ljudi ali strojev) za opravljanje dela s spoštovanjem nabora proceduralnih pravil«. Dumas et al. (2005, str. 22) dodajajo, da se delovni tok osredotoča na strukturo delovnega procesa in ne na samo vsebino posameznih aktivnosti, na drugi strani pa management delovnih tokov služi tudi kot vez med uporabniki (ljudmi) in poslovnimi programskimi rešitvami, ki so potrebne za izvrševanje posameznih nalog. Organizacija Workflow Management Coalition, (1996) trdi, da sta pojma delovni tok (angl. *workflow*) in management delovnih tokov (angl. *workflow management*) sinonima.

Na podlagi opisanih definicij pojma delovni tok je razvidno, da obstajajo različne uporabe, ki med seboj niso skladne, vseeno pa menimo, da je za boljše razumevanje področja in te magistrske naloge primernejša sledeča kategorizacija:

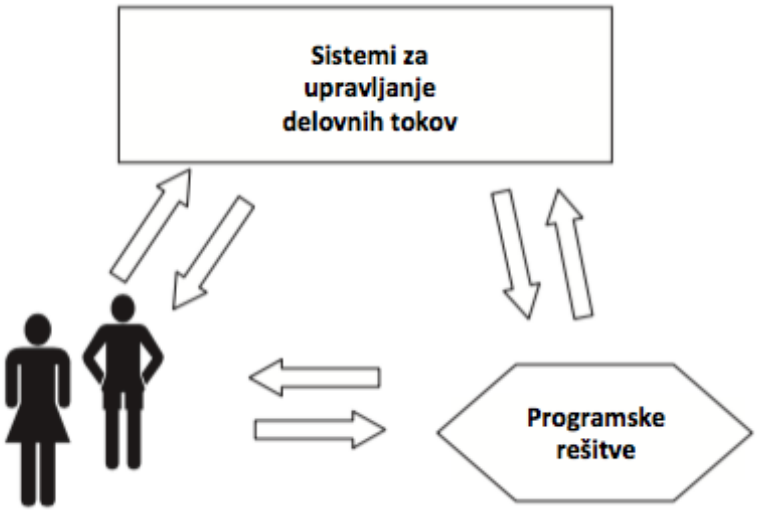
- delovni tok: avtomatizacija poslovnih procesov, v celoti ali delno, pri kateri gre za posredovanje informacij in/ali dokumentov od enega uporabnika do drugega za opravljanje dela s spoštovanjem nabora proceduralnih pravil;

- management delovnih tokov: upravljanje avtomatizacije poslovnih procesov, v celoti ali delno, pri kateri gre za posredovanje informacij in/ali dokumentov od enega uporabnika do drugega (ljudi ali strojev) za opravljanje dela s spoštovanjem nabora proceduralnih pravil.

1.2.2 Management delovnih tokov

Na podlagi in za potrebe managementa delovnih tokov so bili razviti sistemi za management delovnih tokov (angl. *Workflow Management Systems*). Organizacija Workflow Management Coalition (1996) opredeljuje sistem za management delovnih tokov kot sistem, ki definira, ustvarja in upravlja izvajanje delovnih tokov s pomočjo programskih rešitev, ki temeljijo na enem ali več izvajalnikih delovnega toka. Ta je sposoben interpretacije poslovnih procesov, interakcije z uporabniki delovnega toka ter interakcije in uporabe programskih rešitev. Umestitev sistemov za management delovnih tokov je prikazana na Sliki 5.

Slika 5: Delovni tok kot povezava med programskimi rešitvami in uporabniki.



Vir: M. Dumas et al., Process-Aware Information Systems, 2005, str. 23.

Sistem za management delovnih tokov podpira nabor poslovnih procesov tako, da s pomočjo izvajalnika delovnih tokov izvaja poslovne procese na podlagi procesnih specifikacij. Procesna specifikacija opisuje tip procesa, ki bo izveden v posamezni instanci delovnega toka. Vsak delovni tok temelji na organizacijski instanci, ki je odgovorna za specifikacije sheme delovnih tokov in na podlagi katere izvajalnik delovnega toka sprejema odločitve o izvajanju delovnega toka. Vsaka instanca delovnega toka mora imeti nadrejeno organizacijsko instanco, ki je odgovorna za izvajanje (Dumas et al., 2005, str. 24).

Sami sistemi za management delovnih tokov pa omogočajo tudi strukturne spremembe vseh vrst na delovnih tokovih, brez da bi bilo potrebno zahtevno poseganje v programsko kodo programske rešitve. Cilj uporabe sistema za management delovnih tokov je poenostavitev razvoja poslovnih programskih rešitev, saj omogoča tudi ponovno uporabo posameznih komponent programskih rešitev, saj nekateri deli procesov zahtevajo ponovno uprizoritev v različnih programskih rešitvah (Dumas et al., 2005, str. 24).

Za delovanje sistema za management poslovnih programskih rešitev je ključen izvajalnik delovnega toka (angl. *workflow engine*). Osnovna naloga izvajalnika je razumevanje opisov delovnih tokov, ki so predstavljeni v različnih notacijah ali programskih jezikih, hkrati pa služi tudi za ažurno uvedbo sprememb na delovnem toku v samo izvajanje (Dumas et al., 2005, str. 23).

1.3 Temeljne razlike med pojmoma

Če povzamemo, da je:

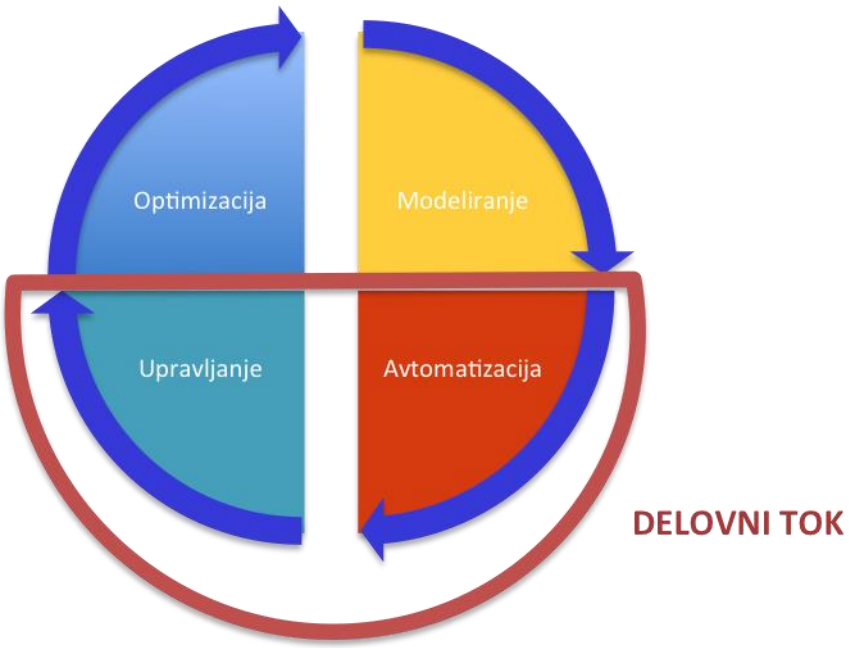
- poslovni proces v grobem skupek logično organiziranih zaporednih in/ali vzporednih aktivnosti, ki jih kot odziv na vhodni vložek izvajajo ljudje ali programske rešitve z namenom doseganja želenih rezultatov (izložkov) znotraj ene ali več organizacij;

- MPP je poslovni pristop k upravljanju sprememb poslovnih procesov. Predstavlja mnogo širše področje obravnave kot pri prenovi poslovnih procesov (angl. *Business Process Reengineering* – BPR). Usmerjen je v poslovno povezovanje procesov s procesi poslovnih partnerjev in z njihovimi informacijskimi sistemi, medtem ko je

- delovni tok v literaturi predstavljen kot avtomatizacija poslovnih procesov, v celoti ali delno, pri kateri gre za posredovanje informacij in/ali dokumentov od enega uporabnika do drugega, za opravljanje dela, s spoštovanjem nabora proceduralnih pravil.

Iz navedenih teoretičnih spoznanj je torej razvidno, da med pojmoma obstaja večja razlika. Iz navedenih definicij je očitno, da je delovni tok podmnožica MPP, gre za avtomatizacijo poslovnega procesa. Poudariti je potrebno, da na področju pojma delovni tok prevladuje zmeda in hkrati tudi zloraba pojma, ki se dogaja na trgu orodij, kot poudarja tudi podjetje Microsoft na svoji spletni strani (Workflow and Process, 2016). Microsoft v nadaljevanju prispevka pojasni, da je delovni tok v sklopu MPP, sinonim za orkestracijo poslovnih procesov. Skrbi za izvajanje in nadzor izvajanja, ustvarjanje dogodkov in ostalih dolžnosti, ki so potrebne za pravilno izvedbo procesa. Delovni tok, ki je izvršljiv poslovni proces, zajema vse elemente, ki so relevantni pri avtomatizaciji procesa. Zajema posamezne aktivnosti, njihov logični vrstni red izvajanja, podatke in informacije, ki vstopajo in izstopajo iz procesa in aktivnosti, dokumente in način, kako so ostali viri, ki sodelujejo v procesu, udeleženi (Ter Hofstede, van der Aalst, Adams & Russel, 2010, str. 3). Mesto delovnega toka v sklopu MPP je prikazano na Sliki 6.

Slika 6: Umestitev delovnega toka v MPP



Vir: Prirejeno po M. K. Strupe, Workflow Automation – Remembering Where BPM Came From, 2010.

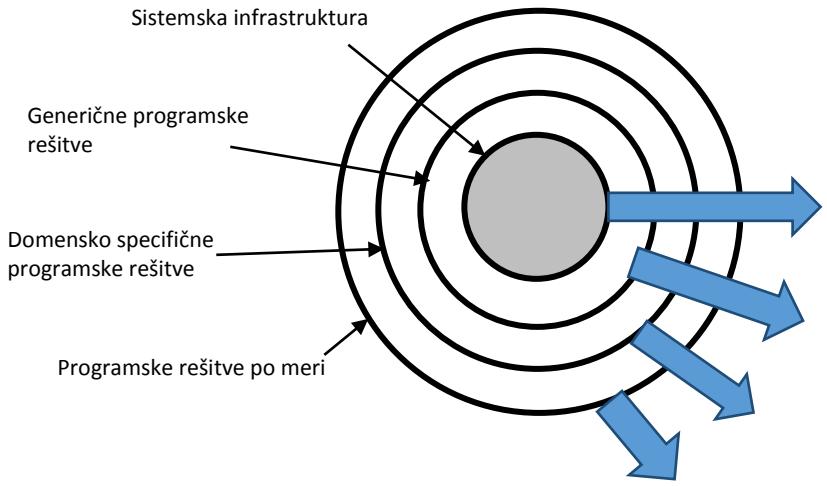
2 RAZVOJNA OGRODJA, POSLOVNI PROCESI IN JEDRNE TEHNOLOGIJE

Eden izmed ključnih izzivov današnjih organizacij je predelava idej in konceptov v inovativne proizvode in storitve v okolju, ki deluje in se spreminja vse hitreje. Razvoj internetnih tehnologij je pripeljal do situacije, ko so se začele razvijati t. i. virtualne organizacije. Organizacije, ki so razkropljene po celem svetu, po različnih časovnih pasovih in obvladajo razne veščine, so s pojavom internetnih tehnologij začele izkoriščati sinergije, ki ob povezavi s partnerji nastanejo. Podjetja in posamezniki iz različnih delov sveta so se virtualno združili v eno virtualno organizacijo, kjer vsak deležnik opravlja svoj nabor poslovnih procesov, potrebnih za delovanje virtualne organizacije. Svoje lastne poslovne procese so lahko integrirali v poslovne procese ostalih članov, kar je sprožilo val trendov, ki so izoblikovali moderne tehnologije za gradnjo poslovnih programskih rešitev (Dumas et al., 2005, str. 4).

Slika 7 prikazuje ključne razvojne trende na področju poslovnih programskih rešitev, ki so se kasneje razvili v plasti sodobnih informacijskih sistemov. V centru je prikazana sistemska infrastruktura, ki je sestavljena iz strojne opreme in operacijskega sistema, druga plast predstavlja generične programske rešitve, ki se uporabljajo množično, v večini organizacij ter tudi v sklopu organizacije v večini oddelkov. Primer takšne programske rešitve so sistemi za upravljanje baz podatkov (DBMS), pisarniška orodja (primer Microsoft Office) in podobni. Tretja plast je sestavljena iz domensko specifičnih poslovnih programskih rešitev, to so rešitve, ki se uporabljajo samo v organizacijah oz. oddelkih s specifično funkcijo, primer takšnih rešitev so orodja CAD (angl. *Computer Aided Development*), orodja za podporo odločanju,

računovodske programske rešitve ipd. Četrta plast pa predstavlja programske rešitve po meri, razvite za potrebe posamezne organizacije (Dumas et al., 2005, str. 4).

Slika 7: Trendi razvoja poslovnih programskih rešitev v povezavi z MPP skozi čas



Vir: M. Dumas et al., Process-Aware Information Systems, 2005, str. 4.

V 60. letih preteklega stoletja druga in tretja plast nista bili še poznani, nista obstajali, informacijski sistemi so bili grajeni na sloju operacijskih sistemov z omejeno funkcionalnostjo. Glede na to, da nobena domensko specifična in generična programska rešitev ni obstajala, so takrat poslovne programske rešitve bile le tiste, ki so bile razvite po meri, torej sta obstajali le prva in četrta plast na Sliki 1, druga in tretja plast pa sta se razvili kasneje. Današnji trend razvoja pa je takšen, kot prikazujejo modre puščice na Sliki 1, vsaka plast postaja debelejša, medtem ko pridobiva nove funkcionalnosti. Današnji operacijski sistemi nudijo, omogočajo, veliko več funkcij kot tedanji, posebej na področju omrežnih tehnologij. DBMS-sistemi danes združujejo funkcionalnosti, ki so v preteklosti bile vgrajene v domensko specifične poslovne programske rešitve in poslovne programske rešitve po meri. Kompleksnost in število raznih domensko specifičnih in po meri programskih rešitev se je drastično povečalo zaradi večanja števila različnih uporabnikov in prilagajanja njihovim potrebam. Pojav internetnih tehnologij je od organizacij zahteval ter hkrati omogočil, da so programske rešitve na daljavo dostopne tudi njihovim poslovnim partnerjem in kupcem. Širitev spektra uporabnikov in nalog, ki jih je potrebno in možno informacijsko podpreti, je pripeljala do sprememb v načinu pristopa k razvoju programske opreme. Usmeritev oz. osredotočenost k programiranju pri razvoju poslovnih programskih rešitev je zamenjala usmeritev k integraciji poslovnih programskih rešitev, ta je sedaj ključni fokus pri informatizaciji poslovanja. Usmeritev k integraciji poslovnih programskih rešitev razvijalcem predstavlja pomembnejši izziv kot samo programiranje posameznih poslovnih programskih rešitev, ključna sta orkestracija delovanja ter optimizacija komunikacijskih povezav med posameznimi programskimi rešitvami ter posameznimi plastmi informacijskih sistemov (Dumas et al., 2005, str. 6).

Vzporedno s trendom »od programiranja k montaži« se je na področju razvoja poslovnih programskih rešitev razvil še en trend. To je zasuk iz »podatkovne usmeritve programskih rešitev« proti »procesni usmeritvi programskih rešitev«. V 70. in 80. letih je bilo ključno vodilo pri razvoju podatkovna usmerjenost oz. pristop, glavni izzivi informacijske tehnologije so bili shranjevanje, pridobivanje in predstavitev podatkov, posledično je bilo modeliranje podatkovne sheme najbolj kritično in začetno opravilo pri razvoju poslovnih programskih rešitev. Omenjeni pristop je pripeljal do razvoja zmogljivih tehnologij in orodij za gradnjo podatkovno usmerjenih informacijskih sistemov. Modeliranje poslovnih procesov je tedaj bilo pogosto zapostavljeno. Posledica tega je bila razpršenost posameznih poslovnih procesov in njihove logike med posameznimi programskimi rešitvami in ročnimi procedurami, to pa je zelo oteževalo in v nekaterih situacijah tudi onemogočalo optimizacijo poslovnih procesov in njihovo spreminjanje, saj bi to pomenilo tudi potrebo po ponovnem razvoju programske opreme. Veliko je bilo tud primerov, ko so bili poslovni procesi prilagojeni programskim rešitvam in ne obratno, kar je v večini primerov za organizacijo pomenilo neučinkovito delovanje, veliko potreb po ročnih opravilih, težave pri delitvi odgovornosti, težave pri iskanju ozkih grl, nepotrebna grupiranja opravil ter odvečne podatkovne vnose. Managerski trendi v 90. letih, kot na primer reinženiring poslovnih procesov (BPR), so v stroko doprinesli nekoliko več poudarka in osredotočenja na procese. Rezultat tega je procesno usmerjen pristop k razvoju poslovnih programskih rešitev (Manouvrier & Menard, 2007, str. 9–11).

Hkrati pa se je z razvojem metod in pristopov k razvoju poslovnih programskih rešitev pojavila še dodatna sprememba – klasično načrtovanje informacijskih sistemov je nadomestilo preoblikovanje informacijskih sistemov. Zaradi množične uporabe internetnih tehnologij so informacijski sistemi prisiljeni k hitrim prilagajanjem v organizacijskih poslovnih procesih. To pomeni, da razvoj novih informacijskih sistemov ni več tako pogost kot pred časom, temveč se poslovne programske rešitve preuredijo, nadgradijo, ali se za novo programsko rešitev uporabi nekatere »sestavne dele« starejše programske rešitve. Poslovne programske rešitve so v današnjem času večinoma sestavljene iz komponent, ki tvorijo celoto, so modularne. Omenjeni pristop omogoča hitro prilagajanje poslovnih programskih rešitev sprememb in potreb v organizaciji, pri poslovnih partnerjih in kupcih, skratka poslovnih procesih. Za obvladovanje tovrstnega razvoja poslovnih programskih rešitev je bil razvit pristop modelsko vodene arhitekture (angl. *Model-Driven Architecture* – MDA). Prednosti, ki jih prinaša omenjeni pristop, so med drugimi tudi samodejno generiranje programske kode na podlagi modela poslovnega procesa in preoblikovanje obstoječe programske kode na podlagi sprememb na procesnem modelu (Manouvrier & Menard, 2007, str. 9–11).

Sinteza omenjenih trendov, ki so se pojavili v procesu razvoja poslovnih programskih rešitev in njihovih tehnologij, so skupaj pripravili teren za procesno usmerjene informacijske sisteme (angl. *Process-Aware Information Systems*). Gre za informacijske sisteme, ki vključujejo ljudi, poslovne programske rešitve in informacijske vire ter jih krmilijo na podlagi poslovnega procesa (Dumas et al., 2005, str. 6).

2.1 Razvojna ogrodja za gradnjo poslovnih programskih rešitev

Pri gradnji programskih rešitev je platforma tista, ki omogoča oz. opredeljuje tehnološko infrastrukturo. Arhitekturne smernice narekujejo krovne okvirje za ureditev elementov v informacijskih sistemih ter interakcijo med temi elementi, platforma pa je tista, ki določa, katere so tehnološke rešitve, tako na strani programske opreme kot strojne opreme. Informacijskemu sistemu dodeljuje orodja in mehanizme, tehnološke rešitve, ki se uporabljajo za razvoj poslovnih programskih rešitev ter izvajanje teh. Obstaja več nivojev, kjer se termin »platforma« uporablja (Kurbel, 2008, str. 122):

- platforma strojne opreme: gre za skupek strojnih komponent, ki skupaj sestavljajo računalnik oz. računalniški sistem, za katerega je bil operacijski sistem narejen in ga podpira. Primeri platform strojne opreme so PC (osebni računalniki) z Intel procesorji in podobnimi, delovne postaje Sun SPARC in podobni ter IBM-ov AS/400 in podobni strežniški sistemi. Vsaka kombinacija strojne opreme za delovanje zahteva kompatibilni operacijski sistem oz. programsko opremo, ki podpira vse strojne komponente in njihovo sodelovanje;

- platforma strojne in programske opreme: sestavljena je iz strojne opreme in sistemske programske opreme (operacijski sistem, mrežne komponente, uporabniški vmesnik ...), ki je prilagojena za delovanje z določeno strojno opremo. Kombinacija strojne in programske opreme narekuje, katere programske rešitve bodo delovale in se izvajale na dotični platformi ter na kakšen način. Primeri takšnih platform so osebni računalniki z operacijskim sistemom Windows in podobnimi, delovne postaje z operacijskim sistemom Solaris in podobnimi, Applovi Macintosh računalniki z lastnim OS X operacijskim sistemom in podobnimi operacijskimi sistemi;

- platforma programske opreme: kombinacija osnovnih programskih komponent, ki določajo, kako bodo ostale programske rešitve delovale in kako bodo razvite, katere tehnologije bodo potrebne za razvoj, izvedbo in uporabo s strani uporabnikov. Programska platforma za delovanje potrebuje strojno opremo, vendar ni odvisna od kombinacije strojne opreme. Primer platforme programske opreme je Javanska platforma, ki deluje na več različnih kombinacijah strojne in programske opreme.

Termin »platforma« pa se ne uporablja samo na področju programske in strojne opreme, ampak se na področju informatike uporablja na več različnih področjih. Aplikacijski strežnik se pogosto v strokovni literaturi imenuje platforma, ker predstavlja osnovno infrastrukturo za razvoj in delovanje mrežnih večuporabniških programskih rešitev. Java EE (Java Enterprise Edition), skupek višjenivojskih Java komponent, je platforma, ker združuje funkcionalnosti za razvoj in izvajanje poslovnih programskih rešitev. Takšen primer platforme je obravnavan v tem razdelku magistrske naloge, za poenostavitev pojmov pa bomo platforme poimenovali s sopomenko »razvojno ogrodje« (Kurbel, 2008, str. 123).

Pri gradnji poslovnih programskih rešitev so razvojna ogrodja za razvoj poslovne programske opreme ključnega pomena. Uporabljena razvojna ogrodja narekujejo, katere tehnologije bodo pri razvoju, izvajanju in uporabi poslovnih programskih rešitev uporabljene, ter katere omejitve bodo pri razvoju in izvajanju upoštevane in prisotne. Posebej uporabljeno razvojno ogrodje določa, kakšna bo arhitektura informacijskega sistema, katera razvojna orodja bodo uporabljena ter kakšne bodo možnosti nadgradnje informacijskega sistema (Kurbel, 2008, str. 123).

Kljub temu, da obstaja mnogo razvojnih ogrodij (veliko je tudi takšnih, ki so že tudi poslovne programske rešitve), so v obravnavanem primeru nekatera, ki so širše uporabljena, saj so prisotna pri informatizaciji poslovanja večine svetovnih organizacij. Razvojna ogrodja, ki omogočajo razvoj poslovnih programskih rešitev, integracijo novih programskih rešitev v obstoječi informacijski sistem ter izvajanje programskih rešitev in so najmnožičneje uporabljena v svetu razvoja poslovnih programskih rešitev, so (Kurbel, 2008, str. 123):

- Javanska platforma;
- Microsoft .NET;
- SAP NetWeaver;
- IBM WebSphere;
- LAMP (Linux, Apache, MySQL, Perl, Python, PHP).

2.2 Jedrne tehnologije v povezavi z razvojnimi ogrodji

Jedrnih tehnologij, ki povezujejo razvojna ogrodja in so pomembna za sodelovanje s poslovnimi procesi, je sicer veliko, v literaturi pa velja prepričanje, da je ključna tehnologija, ki je najbolj zaslužna za uspeh pri povezavi poslovnih procesov s poslovnimi programskimi rešitvami, paradigma storitveno orientirane arhitekture (v nadaljevanju SOA), ta namreč omogoča povezovanje enostavnih funkcij, storitev, v kompleksnejše storitve, ključna lastnost SOA-e pa je ta, da je lahko vsaka od storitev, ki med seboj sodelujejo, razvita z uporabo drugih tehnologij, v različnih programskih jezikih (SOA bi sicer lahko spadala v katero drugo poglavje, saj je to paradigma, ki ni povezana le z obravnavano tematiko temveč z gradnjo poslovnih programskih rešitev nasploh, pa vendar sta SOA in PAIS "rastla" skupaj. To je razlog, da jo v nalogi umeščamo v ta razdelek, ker je obravnavana z vidika razvojnih ogrodij in razvoja poslovnih programskih rešitev, ki temeljijo na poslovnih procesih in na njih temelječih programskih rešitvah). Poleg tega pa so z razvojnimi ogrodji povezane vse tehnologije, ki omogočajo vzpostavitev arhitekture poslovne programske rešitve ter njeno brezhibno izvajanje.

»Struktura je pomembna« trdijo Bass, Clements in Kazman (2003, str. 44). Arhitektura ali struktura definira elemente sistema, njihovo vlogo in medsebojne povezave. Vsak sistem ima svojo arhitekturo, tako tudi sistemi, ki tvorijo informacijsko tehnologijo (takšni in drugačni računalniki). Računalnik ima svojo arhitekturo strojne opreme, svojo arhitekturo pa imajo tudi programske rešitve. Booch (2006, str. 9–11) pravi, da je arhitektura programske opreme namerna, če je bila eksplicitno identificirana in implementirana, nenamerna arhitektura pa izhaja

iz individualnih gradenj sistemov, ki se dogajajo med razvojem programskih rešitev. Arhitektura določa, kako poslovni procesi, podatki, programske rešitve in tehnologije sodelujejo (Wanted: Enterprise Architects, 2006), dobra arhitektura pa je tista, ki ima sledeče kvalitete:

- vzdržljivost;
- stabilnost;
- fleksibilnost.

Iz tega izhaja, da so ključne jedrne tehnologije v povezavi z razvojnimi ogrodji tiste, ki tvorijo in podpirajo stabilno, trdno in fleksibilno arhitekturo celotnega informacijskega sistema.

2.2.1 Arhitektura modernih razvojnih ogrodij

Že od nekdaj se arhitekture programske opreme gradijo v plasteh, aktualna arhitektura programske opreme pa je t. i. večplastna arhitektura, v literaturi pa se pojavlja tudi izraz n-plastna arhitektura (N-Tier Data Applications Overview, 2016). Večplastna arhitektura se je razvila iz triplastne arhitekture, ki je bila aktualna v 90. letih prejšnjega stoletja. Programske rešitve so bile namreč zgrajene iz treh plasti: predstavitvena plast (angl. *presentation tier*), poslovna plast (angl. *business tier*) in podatkovna plast (angl. *data tier*). Vse pa se je spremenilo s prihodom elektronskega načina poslovanja, pojavila se je potreba po dostopu do informacijskega sistema posamezne organizacije z uporabo spletnega brskalnika, kar je zahtevalo vgraditev dodatne plasti, spletnega strežnika (Kurbel, 2008, str. 104).

Tako se je pojavila večplastna arhitektura (angl. *multi-tier architecture*), ki je sčasoma poleg spletnega strežnika za dostopanje do informacijskega sistema z uporabo spletnega brskalnika začela vsebovati tudi druge plasti kot npr. tehnološke rešitve za implementacijo mobilnih tehnologij. Posebnost večplastnih arhitektur je razporeditev procesnih zahtev med uporabnika in strežnik. Osnovna oblika večplastne arhitekture je zgrajena iz treh osnovnih plasti, ki so posledično lahko zgrajene iz podplasti (N-Tier Data Applications Overview, 2016; Application Layers, 2016):

- predstavitvena plast: omogoča interakcijo uporabnikov s programskimi rešitvami, večkrat pa vsebuje tudi dodatno programsko logiko, kot so komponente za zajem podatkov in objekti za prikaz podatkov. Najpomembnejše tehnologije so:
    - Javanska platforma: Java Virtual Machine, JavaServer Faces, JavaServer Pages;
    - .NET ogrodje: Windows Presentation Foundation, Microsoft Windows Forms, ASP.NET Web Forms;
- srednja plast je tista, ki je glavni prevodnik podatkov in informacij med predstavitveno plastjo in podatkovno plastjo, deli pa se običajno na tri podplasti:

- podplast s poslovno logiko (angl. *business logic layer*), ki skrbi za spoštovanje poslovnih pravil in preverjanje ter potrjevanje podatkov;
- podplast za dostop do podatkov (angl. *data access layer*), ki skrbi za pravilno predstavitev podatkov;
- podplast za splošne programske storitve (angl. *common application services*), kot je avtorizacija in avtentikacija uporabnikov.

Najpomembnejše tehnologije srednje plasti so:

- Javanska platforma: Java Message Service, Enterprise JavaBeans, Java Persistence API, Business Process Execution Language;
- .NET ogrodje: Windows Communication Foundation, Enterprise Application Blocks, Windows Workflow Foundation;

- podatkovno plast sestavljajo podatkovni strežniki z bazami podatkov, ki shranjujejo podatke, potrebne za delovanje poslovnih programskih rešitev, dostopna pa je le iz srednje plasti. Najpomembnejše tehnologije podatkovne plasti so:

  - Javanska platforma: Oracle Enterprise Server, IBM DB2, SAP Sysbase ASE;
  - Microsoft: Microsoft SQL Server.

2.2.2 Storitveno usmerjena arhitektura – SOA

Zaradi narave razvoja poslovnih programskih rešitev se pri prenovi in nadgradnji poslovnih programskih rešitev pogosto zgodi, da dve ali več programskih rešitev med seboj niso kompatibilne oz. je sodelovanje (izmenjava podatkov) med njimi oteženo oz. nemogoče. Do tega pride, ker so se poslovne programske rešitve nekoč gradile t. i. silosno, zelo ločene ena od druge, brez medsebojne integracije. V primeru poslovnih programskih rešitev, ki temeljijo na poslovnih procesih, obstaja tesna povezava med integracijskimi tehnologijami in posameznimi poslovnimi programskimi rešitvami. Kadarkoli pride do spremembe poslovnega procesa, se posledično zgodi tudi sprememba v integracijskih tehnologijah, ki omogočajo integracijo poslovne programske rešitve v sistem organizacije. Posledica tega je povišanje operativnih stroškov ter čas, potreben za izvedbo integracije (Weske, 2012, str. 27).

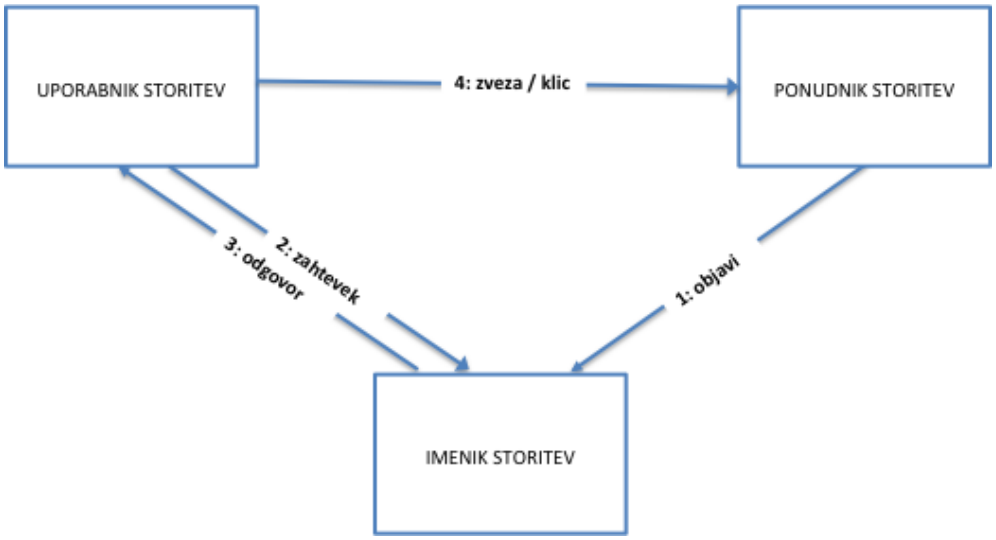
Pogost pojav pri poslovnih procesih je ta, da so podprti s strani več programskih rešitev, oz. poteka izvajanje poslovnega procesa skozi več aplikacij. Kadar pride do sprememb v poslovnem procesu, lahko to zahteva prilagoditve v več kot samo eni programski rešitvi, kar še dodatno poviša stroške vzdrževanja takšnih programskih rešitev (Behara, 2006, str. 2).

Storitveno usmerjena arhitektura je računalniška programska arhitektura, ki ustvarja okolje za opis, ponudbo in uporabo programskih storitev, servisov (Weske, 2012, str. 58). Jurič in Pant (2008, str. 6) v svoji knjigi omenjata, da je storitveno orientirana arhitektura – SOA (angl. *Service-Oriented Architecture*) najpogosteje omenjena paradigma, ko je obravnavana tematika

področje managementa poslovnih procesov. Sodobne poslovne programske rešitve so lahko zgrajene iz več heterogenih aplikacij, ki med seboj komunicirajo preko komunikacijskih omrežji, kot sta lokalno omrežje (LAN) in svetovni splet (HTTP).

Storitve v osnovi zagotavljajo opravljanje določenih funkcij, lahko je to samo ena funkcija, lahko jih je več, funkcije so lahko enostavnejše in tudi bolj zapletene. Paradigma SOA sama omogoča povezovanje enostavnih funkcij, storitev, v kompleksnejše storitve, ključna lastnost SOA-e pa je ta, da je lahko vsaka od storitev, ki med seboj sodelujejo, razvita z uporabo drugih tehnologij (večnivojska arhitektura, odjemalec – strežnik, itd.), v različnih programskih jezikih (C#, Visual Basic, Java, ipd.) ter se lahko izvaja na različnih strežniških arhitekturah (Jurič & Pant, 2008, str. 6).

Slika 8: Vloge v storitveno orientirani arhitekturi



Vir: M. Weske, Business Process Management Concepts, Languages Architectures (2nd ed.), 2012, str. 59.

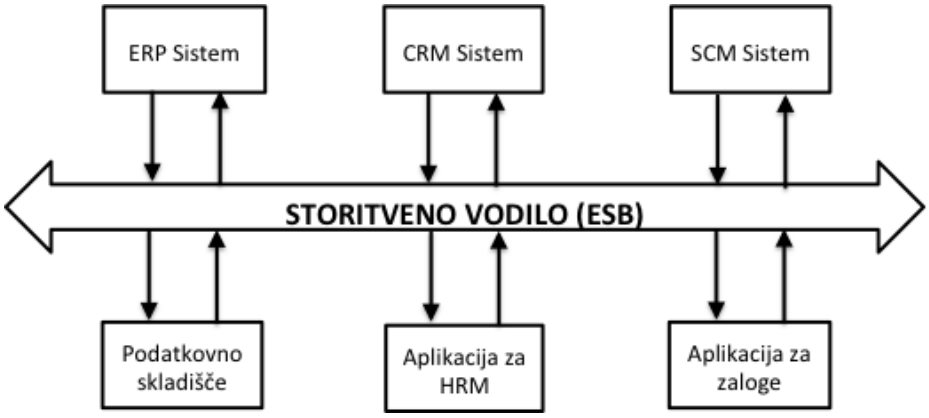
Samo razumevanje pojma storitve se pri tehnologiji SOA ne ustavi pri storitvah, ki jih razumejo računalniški sistemi, temveč gre za razumevanje storitve, kot to razumemo v realnem življenju. Za boljšo predstavo lahko navedemo na primer storitev popravila vozila. Storitev, ki jo opravlja servisna delavnica, mora biti predstavljena na način, da jo uporabnik storitev, v tem primeru voznik vozila, razume in posledično uporablja. Voznik pripelje vozilo na popravilo ter, ko je vozilo s strani delavnice popravljeno, plača račun za opravljene storitve in vozilo prevzame, storitev je tako zaključena. Tovrstne in ostale storitve, ki jih uporabniki storitev potrebujejo, lahko najdejo v imenikih, kot so npr. rumene strani, na podoben način deluje logika storitveno usmerjene arhitekture. Primer vlog, prisotnih pri SOA je prikazan na Sliki 8 (Weske, 2012, str. 57). Tovrstna ideja stoji za storitveno usmerjeno arhitekturo, ki jo obravnavamo v tem razdelku, vloge, ki so povezane z uporabo raznoraznih storitev, kot je opisano v prejšnjem odstavku (imenik storitev, uporabnik in ponudnik), morajo biti prisotne tudi pri SOA (Weske, 2012, str. 57).

Povzamemo lahko, da je SOA prinesla nov standard razvoja in uvedbe poslovnih programskih rešitev. Razvijalci in vzdrževalci poslovne programske opreme so pred uvedbo SOA-e imeli velike težave takrat, ko so razvijali nove programske rešitve, ki so morale sodelovati z obstoječimi. Za vsako programsko rešitev je bilo potrebno pripraviti nov vtičnik, ki bo razumel delovanje preostalih, dodatne težave pa so predstavljale rešitve, ki so bile razvite z uporabo različnih programskih jezikov. Cilj uvedbe SOA-e je bil ravno ta, da se razvijalci in vzdrževalci izognejo opisanim težavam, kar naj bi znižalo stroške razvoja in vzdrževanja programskih rešitev, rezultate pa naj bi opazili tudi uporabniki, tako iz uporabniškega vidika kot tudi iz stroškovnega. Kot že opisano, z uvedbo SOA-e se uvedejo storitve, posamezne avtomatizirane aktivnosti poslovnih procesov izvajajo storitveni servisi, to je poslovna programska rešitev, ki opravi storitev, ki jo aktivnost zahteva (Jurič & Pant, 2008, str. 16).

Cilj SOA integrirati funkcionalnosti storitve v nove kontekste izvajanja, v poslovne procese, se da uresničiti s kompozicijo storitev, to predstavlja neke vrste lepilo med storitvami na eni in poslovnimi procesi na drugi strani. Vidik kompozicije namreč upošteva poslovne procese kot ključni sestavni del, podobno kot objektni jeziki upoštevajo objekte. Za uspešno kompozicijo pa je poleg kompozicije potrebno omeniti, da je ključnega pomena tudi način dostopanja do storitev, trenutno najučinkovitejši je način dostopa z uporabo spletnih storitev, posebno vlogo pri tem pa ima osnovni sestavni del arhitekture SOA, storitveno vodilo (angl. *Enterprise Service Bus*), v nadaljevanju ESB (Slika 9), ki zagotavlja (Jurič, 2005):

- upravljanje varnosti, korelacij, transakcij in koordinacije spletnih storitev na enoten način ter
- povezljivost med raznimi spletnimi storitvami.

Slika 9: Storitveno vodilo (ESB)



Vir: M. Weske, Business Process Management Concepts, Languages Architectures (2nd ed.), 2012, str. 64.

ESB pri arhitekturi SOA deluje kot vmesni sloj, preko katerega se omogoča dostop množici storitev, ki so del informacijskega sistema organizacije. ESB zagotavlja povezljivost, nadzor in upravljanje s storitvami. To je centralizirano storitveno ogrodje, ki z uporabo različnih protokolov ponuja transparentnost komunikacije med storitvami, to pa pomeni dodano vrednost

v obliki podpore varnosti, učinkovitejših transakcij, dostave sporočil, transformacije, usmerjanja in podobnih storitev (Jurič, 2005).

V arhitekturi SOA je povezovanje storitev med seboj mogoče na dva načina, to sta (Jurič, 2005):

- **orkestracija:** temelji na osnovnem procesu, ki je odgovoren za nemoteno in pravilno delovanje celotnega poslovnega procesa, to pomeni, da se posamezne storitve, ki so med seboj nepovezane, izvajajo v pravilnem vrstnem redu;
- **koreografija:** v nasprotju z orkestracijo se pri koreografiji storitve med seboj same uskladijo, katera je naslednja na vrsti za izvajanje. Med seboj so odvisne ter morajo vedeti, katera se izvaja v naslednjem koraku procesa, za povezavo med njimi pa se uporablja sporočilni sistem. V primeru koreografije gre za manj prožen način v primerjavi z orkestracijo, saj običajno vsaka sprememba posameznega poslovnega procesa zahteva spremembe v večjem številu storitev.

2.3 Jedrne tehnologije v povezavi s poslovnimi procesi

Omeniti je potrebno, da med jedrne tehnologije v povezavi s poslovnimi procesi spadajo tudi pristopi in ostala teoretična spoznanja, ki ne spadajo v sam BPM temveč v pristop k razvoju poslovnih programskih rešitev. Poslovne procese in pripadajoče sodelovanje, izmenjavo informacij in aktivnosti se izraža z uporabo ustreznih tehnologij. V tej nalogi bomo podrobneje obravnavali tehnologiji Windows Workflow Foundation (WF) ter WS-BPEL (Business Process Execution Language), obe bomo tudi podrobno predstavili v naslednjem poglavju, v tem razdelku pa bomo predstavili tehnologije in tehnike modeliranja (notacije), ki se uvrščajo med pomembnejše pri razvoju poslovnih programskih rešitev, ki temeljijo na poslovnih procesih: BPMN (Business Process Modeling Notation), BPML (Business Process Modeling Language), UML (Unified Modeling Language), YAWL (Yet Another Workflow Language) in so navedene na Sliki 10.

Pri kategorizaciji omenjenih programskih jezikov, ki so hkrati tehnologije, je potrebno biti pozoren na kategorizacijo teh. Ionita in Estublier v zbirki Business Process Modeling: Software Engineering, Analysis and Applications (2010, str. 66) predlagata kategorizacijo standardov, jezikov in notacij za modeliranje, avtomatizacijo in izvajanje poslovnih procesov v štirih skupinah:

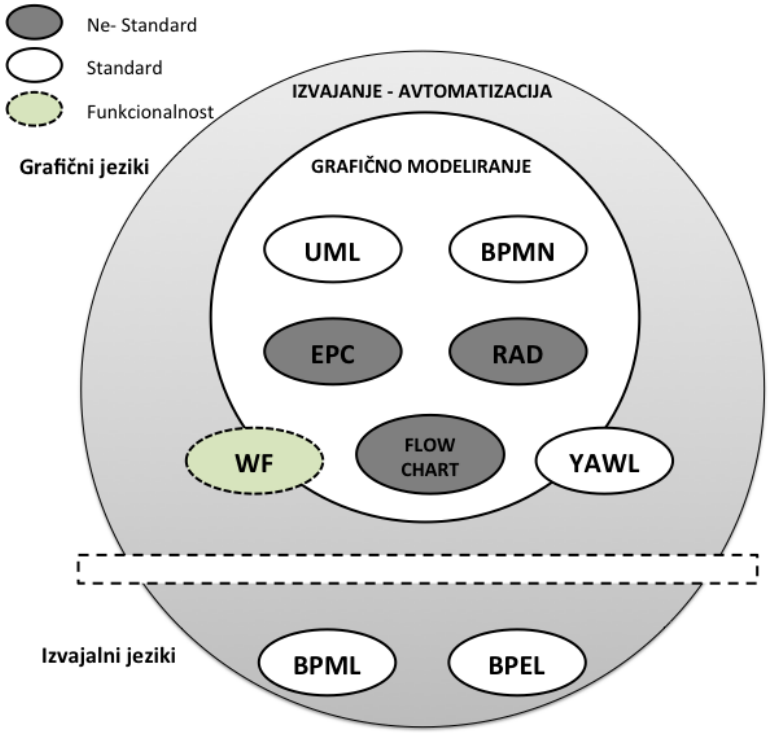
- izvajanje;
- izmenjava;
- grafični prikaz;
- diagnostika.

Za razumevanje področja, ki ga v nalogi obravnavamo, pa sta relevantni le dve izmed naštetih skupin, grafični prikaz modelov poslovnih procesov z visokim nivojem abstrakcije (so razumljivi

poslovnim analitikom) ter standardi, ki so uveljavljeni na področju izvajanja poslovnih procesov. Ionita in Estublier (2010, str. 67) trdita, da je največji izziv današnjih razvijalcev, kako zapolniti vrzel, ki obstaja med tema dvema skupinama. Trenutno najbolj uveljavljeni standardi na obravnavanem področju za pripravo modelov in izvajanje poslovnih procesov so prikazani na Sliki 10 .

Kot je razvidno iz Slike 10, kjer so navedeni najpogosteje uporabljeni izvajalni jeziki in notacije za modeliranje poslovnih procesov, je teh kar nekaj, nekateri so standardizirani, poudariti pa je potrebno, da se Microsoftov Workflow Foundation (WF) v literaturi ne kategorizira kot standard, niti kot programski jezik ali notacija. Singer, Kotrenba in Raß (2014, str. 68–73) in Zapletal, van der Aalst, Russell, Liegl in Werthner (2009, str. 200–209) v svojih prispevkih WF označujejo kot tehnologijo oz. funkcionalnost, ki je del Microsoftovega ogrodja .NET, Zapletal celo meni, da gre za programski vmesnik (angl. *application programming interface* – API).

Slika 10: Pomembnejši jeziki za modeliranje in izvajanje poslovnih procesov



Vir: Povzeto po A. D. Ionita & J. Estublier. Business Process Modeling and Automation with General and Domain Specific Languages, 2010, str. 6; J. Fengel, Business Semantics Alignment for Business Process Model Integration, 2013, str. 117–120.

2.3.1 UML (angl. *Unified Modeling Language*)

Aktivnostni diagrami UML predstavljajo standardizirani grafični programski jezik oz. notacijo, namenjeno modeliranju abstraktnih modelov poslovnih procesov in sistemov, ki ga je predstavila skupina OMG (Object Management Group). Uporabniki lahko UML diagrame prilagodijo svojim potrebam.

Sistemski model je sestavljen iz nabora diagramov, grafičnih vzorcev, ki imajo tudi semantično ozadje. Sistemske modele sestavljene z UML-diagrami je mogoče predstaviti na tri različne načine (Fengel, 2013, str. 102):

- pregled funkcionalnih zahtev: diagrami primerov uporabe (angl. *Use Case*);
- statičen pregled strukture: diagrami razredov (angl. *Class Diagrams*) in kompozitni strukturni diagrami (angl. *Composite Structure Diagrams*);
- dinamični pregled delovanja: zaporedni diagrami (angl. *Sequence Diagrams*) in aktivnostni diagrami (angl. *Activity Diagrams*).

2.3.2 BPMN (angl. *Business Process Management Notation*)

BPMN (Business Process Modeling Notation) je notacija za modeliranje poslovnih procesov. Osnovno poslanstvo BPMN-modelov je omogočanje enostavnejše komunikacije med poslovnimi analitiki pri sprejemanju strateških odločitev, ki se tičejo stroškov, simuliranja scenarijev in podobnih. BPMN modeli pa se uporabljajo tudi kot osnova za specifikacijo programskih potreb ter kot osnova za projekte razvoja poslovnih programskih rešitev. Notacija je bila razvita s strani Business Process Management Initiative (BPMI), trenutno pa vzdrževanje le-te vodi skupina Object Management Group (Xu & de Vrieze, 2013, str. 120).

Modeliranje poslovnih procesov z uporabo BPMN je mogoče z uporabo nabora grafičnih elementov; glavni tipi gradnikov, potrebni za izdelavo modela poslovnega procesa so (Xu & de Vrieze, 2013, str. 120 in Preac, 2011, str. 10):

- objekti za opredelitev toka dogodkov (angl. *Flow Objects*) opredeljujejo obnašanje poslovnih procesov in med katere se uvrščajo:
    - dogodki (angl. *Events*);
    - aktivnosti (angl. *Activities*) in
    - prehodi (angl. *Gateways*);
- povezovalni objekti (angl. *Connecting Objects*), ki se uporabljajo za prikaz vrstnega reda izvajanja toka poslovnih procesov, ter se delijo na:
    - tok sekvenc (angl. *Sequence Flows*);
    - tok sporočil (angl. *Message Flows*) in
    - asociacije (angl. *Associations*);
- organizacijske enote (angl. *Swim Lanes*) imajo funkcijo organizirati aktivnosti poslovnega procesa v ločene kategorije. Z njimi je mogoče prikazati različne funkcionalnosti in odgovornosti, delijo se na:

  - bazene (angl. *Pools*) in
  - steze (angl. *Lanes*);

- artefakti (angl. *Artifacts*) procesu zagotavljajo dodatne informacije, trenutno pa obstajajo trije tipi artefaktov:

  - podatkovni objekti (angl. *Data Objects*);
  - skupina (angl. *Group*) ter
  - tekstovna označba (angl. *Text Annotation*).

Notacija BPMN se je večinoma uporabljala v kombinaciji s programskim jezikom BPEL, s prihodom novejše različice BPMN notacije, ki se poimenuje BPMN 2.0, pa je ta postala tudi izvajalni jezik. Kljub temu da je povezava z BPEL še vedno priljubljena, pa ta ni več nujno potrebna. Različica 2.0 namreč vsebuje tudi spodaj ležeči metamodel in njegovo serializacijo v obliki formata XML (Benedik, 2013, str. 1).

2.3.3 YAWL (angl. *Yet Another Workflow Language*)

Yet Another Workflow Language (YAWL) je bil razvit zaradi pomanjkanja procesnega jezika, ki bi neposredno podpiral vse kontrolne vzorce (angl. *Control-flow patterns*). Osnovni gradniki jezika YAWL so t. i. mreže delovnih tokov (angl. *Workflow nets*). To so podmnožica Petrijevih mrež, ki se od njih razlikujejo v tem, da imajo še dodatna gradnika, ki označujeta vhod (začetek procedure oz. procesa) ter izhod (zaključek procedure oz. procesa), YAWL skoraj v celoti temelji na Petrijevih mrežah z izjemo izvajalne semantike, ki temelji na sistemu prehajanja stanj, za razliko od Petrijevih mrež pa YAWL podpira tudi vse kontrolne vzorce, zaradi česar je bil tudi razvit (Weske, 2012, str. 182).

Pri YAWL gre za odprtokodni sistem, ki poleg samega modelirnega in izvajalnega okolja nudi tudi orodje, ki omogoča modeliranje in vsebuje izvajalnik za izvedbo delovnega toka. Zaradi svoje odprtokodne naravnanosti je uporaba YAWL pogosta v izobraževalnih zavodih po celem svetu, prisoten pa je tudi v informatizaciji poslovnih procesov v nekaterih podjetjih, najodmevnejši primer je UK Telecom (The YAWL Foundation, 2012).

2.3.4 BPML (angl. *Business Process Modeling Language*)

Business Process Modeling Language (BPML) je bil razvit za potrebe modeliranja transakcijskih poslovnih procesov. Gre za označevalni jezik (angl. *markup language*), ki temelji na označevalnem programskem jeziku XML ter ima možnost izvajanja delovnih tokov. Razvit je bil s strani organizacije Business Process Management Initiative (BPMI) – neprofitna organizacija – ki skrbi za promocijo standardov na področju poslovnih procesov. BPML poleg osnovnih in kompleksnejših aktivnosti, konektorjev in dogodkov ponuja tudi možnosti za upravljanje podatkov ob izvajanju poslovnih procesov.

Jezik BPML je bil s strani BPMI razvit istočasno kot notacija BPMN z namenom, da bi bila jezika komplementarna, predstavljala naj bi dva različna nivoja abstrakcije. BPMN, ki predstavlja grafično notacijo, naj bi služil modeliranju poslovnih modelov in poslovnih procesov in ne bi podpiral funkcionalnosti, ki so potrebne za avtomatizacijo aktivnosti, BPML pa bi služil ravno temu, prevajanju BPMN-diagramov v razumljiv označevalni BPML-jezik in avtomatizaciji, izvajanju teh poslovnih procesov (Dumas et al., 2005, str. 283).

3 PREGLED ZNAČILNOSTI DVEH KONKURENČNIH JEDRNIH TEHNOLOGIJ

3.1 .NET in Windows Workflow Foundation

3.1.1 .NET

Microsoft je svojo platformo .NET predstavil leta 2000, takrat so .NET sprva poimenovali "iniciativa" (angl. *initiative*). Uporabniki oz. razvijalci v začetku niso bili prepričani, ali je to produkt, arhitekturna informacijska rešitev ali razvojno ogrodje. Microsoft je namreč pod okrilje .NET blagovne znamke vključil nabor lastnih tehnologij za gradnjo in razvoj programskih rešitev, nekaterih novih ter drugih nadgrajenih različic, ki so bile pred nastopom .NET ogrodja oz. platforme prisotne vsaka pod svojim imenom. Iz tehničnega vidika je v literaturi .NET predstavljen kot platforma, ker omogoča uporabo nabora orodij, mehanizmov in tehnologij za razvoj programskih rešitev ter njihovo izvajanje (Kurbel, 2008, str. 136).

Ob predstavitvi .NET ogrodja je bil cilj podjetja Microsoft razvijalcem ponuditi razvojno ogrodje, ki bo omogočalo celovito paleto storitev, od začetne zasnove programskih rešitev do razvoja in izvajanja, ciljali so na uporabo v poslovne namene, za pripravo poslovnih programskih rešitev in internetnih informacijskih sistemov. Ključni cilji razvoja .NET ogrodja so bili omogočiti (Kurbel, 2008, str. 137):

- porazdeljeno računalništvo in poenostavitev razvoja sistemov "odjemalec/strežnik" in podobnih, ki temeljijo na odprtokodnih internetnih tehnologijah in standardih, kot so HTTP, XML, SOAP, ipd.;
- razvoj poslovnih programskih sistemov in informacijskih sistemov v komponentah, da se omogoči ponovna uporaba in modularna gradnja programskih rešitev na enostavnejši način kot dotlej;
- internetno interoperabilnost; poseben poudarek na programski opremi, ki je nameščena na strežnikih, dostopnih preko internetne povezave (spletne storitve);
- jezikovno neodvisnost; programske rešitve so lahko razvite z uporabo različnih programskih jezikov, a so kljub temu na enostaven način integrirane v obstoječi informacijski sistem ter omogočajo sodelovanje z obstoječimi poslovnimi programskimi rešitvami;

- integracija programskega jezika na programerskem nivoju; v sklopu ene programske rešitve se lahko pri razvoju posamezne sklope programske rešitve razvija v različnih programskih jezikih;
- zanesljivost; programske rešitve, ki delujejo v ogrodju .NET imajo v samem delovanju manjše število napak kot ostale tradicionalne programske rešitve;
- višja varnost z vgrajeno varnostno infrastrukturo. Varnost je pri razvoju programskih rešitev, posebej poslovnih in internetnih rešitev, vedno pereča težava, saj je izpostavljenost informacijskih sistemov in računalnikov internetnim povezavam vedno večja.

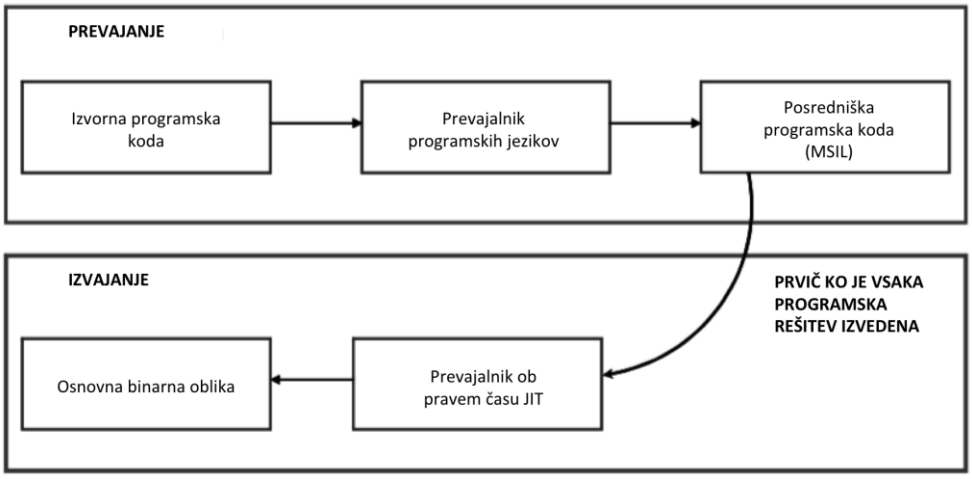
Platformo .NET predstavljajo štirje ključni sestavni deli oz. komponente (Kurbel, 2008, str. 137):

- ogrodje .NET;
- razvojna orodja;
- nabor strežnikov;
- uporabniške programske rešitve.

.NET je temeljno razvojno ogrodje podjetja Microsoft za gradnjo in izvajanje modernih programskih rešitev in spletnih storitev, ki temeljijo na jeziku XML. Uporablja se lahko za razvoj vseh oblik programskih rešitev, od preprostih do najzahtevnejših. Ključne prednosti, ki so po mnenju proizvajalca privlačne za razvijalce pri odločitvi za uporabo razvojnega ogrodja .NET, so samodejno upravljanje z delovnim spominom strojne opreme ter moderni programski jeziki, ki omogočajo učinkovit razvoj tehnološko najnaprednejših programskih rešitev. Ogrodje omogoča okolje t. i. programiranja na visoki ravni (angl. *high-level programming*) z večjim naborom ključnih značilnosti, hkrati pa omogoča dostop do programiranja parametrov in urejanje le-teh tudi na nižji ravni (angl. *low-level programming*). Temeljni cilji .NET razvojnega ogrodja so (Overview of the .NET Framework, 2016):

- zagotavljanje objektno usmerjenega razvojnega okolja ne glede na to, ali je objektna koda shranjena in izvedena lokalno, spletno distribuirana, a izvedena lokalno oz. ali je izvedena na oddaljeni lokaciji;
- zagotavljanje okolja za izvajanje programske kode, ki ne zahteva odvečnih namestitev ter konfliktov med različicami programske rešitve;
- zagotavljanje varnega okolja za izvajanje programske kode, tudi če je ta plod razvoja neznanega ali delno zaupanja vrednega razvijalca;
- uporabniška izkušnja mora biti dobra skozi celotno široko paleto podprtih tipov aplikacij, tako tistih, ki se izvajajo v Windows okolju, kot spletnih aplikacij;
- vsi komunikacijski protokoli in komunikacijske tehnologije, ki jih podpira, temeljijo na standardih; programska koda .NET ogrodja se lahko integrira s kodo drugih ogrodij.

Slika 11: Interpreter (CLR)



Vir: E. K. Kurbel, The Making of Information Systems, 2008, str. 139.

Od samega začetka, torej od različice .NET Framework 1.0, je to razvojno ogrodje sestavljeno iz dveh temeljev, interpreterja (angl. *Common Language Runtime* - CLR) in iz zbirke knjižnic razredov .NET razvojnega ogrodja (angl. *.NET Framework class library*) (Kurbel, 2008, str. 139):

- interpreter (CLR) je temelj ogrodja .NET in skrbi za upravljanje, izvajanje programske kode (angl. *Managed code execution*) ter ob tem zagotavlja jedrne storitve, kot so upravljanje z delovnim spominom računalnika, upravljanje niti (angl. *thread*) ter medprocesno komuniciranje (angl. *.NET remoting*). Interpreter istočasno skrbi tudi za varnost izvajanja programske kode in posledično suverenost izvajalnega okolja in celotnega ogrodja. Bistvo delovanja CLR je prikazano na Sliki 11. Izvorna programska koda, uporabljena ob razvoju programske rešitve s strani razvijalca (angl. *source code*), je s pomočjo jezikovnega prevajalnika (angl. *Language Compiler*) prevedena v psevdokodo, ki se imenuje Microsoft Intermediate Language, v nadaljevanju MSIL. Preden je mogoče MSIL-kodo izvršiti, je ta še enkrat prevedena, tokrat v binarno, osnovno obliko (angl. *native code*), ki jo podpira kombinacija strojne in programske opreme, na kateri se izvaja programska rešitev. Prevajanje MSIL programske kode v binarno obliko se zgodi s pomočjo JIT-prevajalca (angl. *JIT Compiler*), kjer črke JIT pomenijo »Just In Time«, kar v slovenskem jeziku pomeni »ravno ob pravem času«. Že iz samega naziva prevajalnika je razumeti, da se prevod dogaja v času izvajanja programske rešitve;

- zbirka knjižnic razredov je objektno usmerjena zbirka ponovno uporabljivih tipov, ki so namenjeni razvijalcem za kar se da poenostavljen razvoj programskih rešitev. Knjižnice razredov se lahko uporabljajo za razvoj najosnovnejših tekstovnih programskih rešitev kot tudi za razvoj najkompleksnejših, ki temeljijo na najmodernejših tehnologijah.

.NET ogrodje je v obdobju raziskovanja in priprave te magistrske naloge dostopno v več različnih vrstah implementacije, vse oblike pa temeljijo na .NET standardih, ti pa so odvisni od platforme, na kateri se ogrodje in programske rešitve, ki temeljijo na .NET, izvajajo. Razlikujejo se na podlagi oblik programske opreme, tako se ločijo med seboj programske rešitve, namenjene (Overview of the .NET Framework, 2016):

- osebnim računalnikom;
- mobilnim platformam;
- računalniškim igram;
- računalništvu v oblaku.

Hkrati pa se ogrodja vrste .NET razlikujejo še na podlagi platforme:

- Windows;
- Linux;
- iOS;
- Android;
- OS X.

Odprtokodni standardi, ki temeljijo na OSI (Open Source Initiative), so prav tako pomemben gradnik .NET razvojnega ogrodja in hkrati ekosistema.

3.1.2 Windows Workflow Foundation

WF je v literaturi označen kot ogrodje, ki omogoča deklarativno programiranje z uporabo delovnih tokov, in je ključno pri podpori procesov, ki se izvajajo na osnovi dogodkov (angl. *event-driven processes*) in procesov z dolgim časom izvajanja (angl. *long-running processes*). Mnogi WF dojemajo kot izvajalnik (angl. *engine*) za izvajanje delovnih tokov, a je v resnici neke vrste razvojno ogrodje (White, 2013, 19), ki omogoča naravno interpretacijo programske kode, saj jo predstavlja v obliki delovnih tokov (White, 2013, str. 21).

WF namreč omogoča razvoj in izvedbo sistemskih in človeških delovnih tokov ter pretvorbo teh v izvršljivo programsko kodo in posledično v delujoče programske rešitve, ki se izvajajo v operacijskih sistemih družine Microsoft Windows, tako v tistih, namenjenih osebnim računalnikom (Windows XP, Windows 7, Windows 10 ipd.), kot tudi v strežniških operacijskih sistemih (Windows Server 2008, Windows Server 2012, ipd.). Namenjen je razvoju enostavnih aplikacij, kot je prikazovanje interakcijskih gumbov v uporabniškem vmesniku, predvsem pa je namenjen reševanju kompleksnih scenarijev v obsežnih poslovnih programskih rešitvah, kot na primer procesiranje novih naročil, upravljanje s proizvodnim procesom, upravljanje z zalogo materiala, itd. (Windows Workflow Foundation Overview, 2016).

Večina programskih rešitev je razvitih na način, ki omogoča hitro procesiranje informacij, uporabniki preko uporabniških vmesnikov ustvarjajo nove podatke, berejo stare, podatke brišejo ter jih nadgrajujejo, vse to pa se dogaja instantno, v tistem trenutku. Se pa lahko zgodi tudi to, da se določen poslovni proces, da doseže svoj konec, zaključek, izvaja več dni, mesecev, lahko tudi let – to so procesi z dolgim časom izvajanja. Za razvijalce ter tudi za razvojno ogrodje to predstavlja velik izziv, WF pa je namenjen tudi temu, da omogoča upravljanje z delovnim spominom ter hranjenje stanj aktivnosti na učinkovit način, s poudarkom na izvajanju procesov z dolgim časom izvajanja (White, 2013, str. 21).

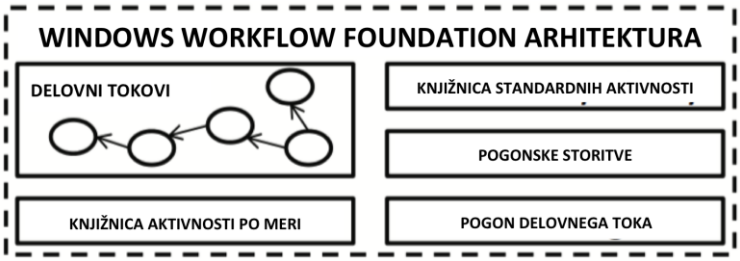
V nasprotju s standardom WS-BPEL, Windows Workflow Foundation (v nadaljevanju WF) ni odprtokodni standard, temveč je standard, razvit s strani podjetja Microsoft kot del razvijalskega ogrodja .NET, trenutno pa ga najdemo v različici ogrodja 4.6. (Lenhard, 2011, str. 13). WF je prišel v uporabo leta 2006 skupaj z različico 3.0 ogrodja .NET, kasneje pa je izboljšana različica bila izdana skupaj z razvijalskim ogrodjem .NET 3.5. V času izdelave te magistrske naloge je aktualna verzija tehnologije Workflow Foundation 4.6, ki je izšla skupaj z .NET 4.6. Microsoft je skupaj z izdajo .NET 4.0 je predstavil tudi popolnoma prenovljeno različico WF z večjimi spremembami v sami strukturi razvijalske tehnologije zaradi začetnih negativnih odzivov na kompleksnost razvoja in gostovanja delovnih tokov s strani razvijalcev orodij. Hkrati pa je bilo tudi razvojno ogrodje .NET 4.0 pomembnejša nadgradnja, želja pa je bila, da bi WF, ki je bil tedaj razmeroma nova tehnologija, lahko bolje izkoristil nove zmogljivosti prenovljenega .NET 4.0 izvajalnika (White, 2013, str. 23).

Pri implementaciji WF kot orkestracijskega jezika v spletnih storitvah, ki temeljijo na SOA paradigmi, je potrebna implementacija tehnologije, ki to omogoča. Pri WF je to mogoče z uporabo tehnologije spletnih storitev Windows Communication Foundation (v nadaljevanju WCF). Omenjena tehnologija pripomore k objavi delovnega toka kot spletne storitve, v terminologiji tehnologije WF pa se to imenuje spletna storitev delovnega toka. (angl. *Workflow Service*) (Lenhard, 2011, str. 14).

Spletna storitev delovnega toka tvori orkestracijo, temeljna razlika med spletno storitvijo delovnega toka in delovnim tokom v klasični obliki pa je, da se v prvem primeru delovni tok prevede in implementira naravnost v obliko spletne storitve, samega delovnega toka pa ni več mogoče uporabiti kot aktivnost (podproces) nekega drugega delovnega toka. V primeru implementacije delovnega toka kot spletne storitve, tehnologija WCF v sodelovanju z WF implementirani delovni tok prevede v spletno storitev z nekaterimi spremembami v programski kodi ter v datotečnem zapisu, ki se v nasprotju s klasičnim XAML zapisom (.xaml) zapiše v obliki (.xamlx), ki je oblika zapisa spletnih storitev (Lenhard, 2011, str. 15).

WCF si skupaj s primerjajočo tehnologijo za gradnjo poslovnih programskih rešitev WS-BPEL deli programski jezik za opisovanje omrežnih storitev (angl. *network services*) WSDL 1.1. Struktura WSDL-ukazov, ki tvorijo vmesnik za upravljanje z delovnimi tokovi, je popolnoma združljiva z WF ukazi, delovni tok oz. programska rešitev, ki na njem temelji, lahko deluje tudi kot spletna storitev (Lenhard, 2011, str. 15). Na Sliki 12 je prikazana arhitektura WF.

Slika 12: Arhitektura WF



Vir: M. Zapletal et al., An Analysis of Windows Workflow's Control-Flow Expressiveness, 2009, str. 200–209.

3.1.2.1 KOMPONENTE WF

WF je, tako kot večina jedrnih tehnologij in ogrodij, sestavljen iz več različnih sestavnih delov oz. komponent in tehnologij, ki so ključne za delovanje in razumevanje delovanja: (White, 2013, str. 26 & Windows Workflow Foundation Overview, 2016)

- aktivnosti;
- izvajalnik delovnega toka (angl. *Workflow Runtime*);
- delovni tokovi;
- orodje za razvoj delovnih tokov (angl. *Workflow Designer*);
- podatkovni model delovnih tokov (angl. *Workflow DataModel*);
- storitve za ohranjanje stanja (angl. *Persistence*);
- nadzor delovanja delovnih tokov (angl. *Monitoring*);
- serializacija (angl. *Serialization*);
- označevalni jezik – XAML (angl. *Markup Language*).

3.1.2.1.1 AKTIVNOSTI

Aktivnosti predstavljajo najosnovnejše gradnike poslovnih procesov, so enote dela, ki se uporabljajo za izvrševanje programske kode v sklopu delovnega toka (White, 2013, str. 27). Sama aktivnost lahko izvaja eno samo operacijo, kot na primer zapis v podatkovno bazo, lahko pa je zgrajena iz drugih aktivnosti. Takšne aktivnosti se lahko poimenuje tudi podprocesi, Microsoft jih imenuje podrejene aktivnosti (angl. *child activities*). Pri WF se aktivnosti razlikujejo po načinu obnašanja, tako obstajajo (Activities Overview, 2016):

- izvajalske (angl. *runtime*) aktivnosti: tako se poimenujejo takrat, ko se aktivnost izvaja;
- oblikovalske (angl. *designtime*) aktivnosti: aktivnost se prikazuje in ureja v grafičnem urejevalniku.

WF ima v svoji osnovi že vgrajene zbirke standardnih aktivnosti (angl. *Base Activity Library*), poleg tega pa lahko sam uporabnik ustvari tudi svoje aktivnosti po meri in tako zmogljivosti WF

razširi, hkrati pa jih lahko tudi ponovno uporabi pri novih delovnih tokovih. Aktivnosti, ki so že sestavni del WF, omogočajo funkcionalnosti, kot je nadzor pretoka (angl. *control flow*), pogojne aktivnosti (angl. *condition*), upravljanje dogodkov (angl. *event handling*), upravljanje stanja (angl. *state management*) ter komuniciranje z drugimi aplikacijami in storitvami.

3.1.2.1.2 IZVAJALNIK DELOVNEGA TOKA

Izvajalnik delovnega toka (angl. *Workflow Runtime*) je, kot že samo ime pove, ključni del, srce tehnologije Workflow Foundation, je sestavni del, ki poganja, izvaja delovne tokove. Naloga izvajalnika delovnega toka je razporejanje in koordiniranje asinhronega delovanja delovnih tokov in aktivnosti znotraj delovnega toka. V primerjavi z WF 3.X, ko je izvajalnik delovnega toka moral delovati kot gostujoči proces izvajalnika ogrodja .NET, se je v različici WF 4.0 in sledečih to spremenilo. Izvajalnik delovnega toka deluje kot samostojna entiteta (White, 2013, str. 26).

3.1.2.1.3 DELOVNI TOKOVI

Delovni tokovi v WF predstavljajo orkestracijo dela na podlagi pripadajočih poslovnih procesov. Mogoče je zgraditi tri različne tipe delovnih tokov, ki podpirajo različne tipe poslovnih procesov oz. zahtev, ki izhajajo iz tipa poslovnega procesa, ki ga razvijalec modelira:

- zaporedni (angl. *sequential*): poslovni procesi po določenih vnaprej začrtanih vzorcih, aktivnosti se izvajajo po vnaprej določenem vrstnem redu, ena za drugo, od začetka do konca. Pot, po kateri se izvajajo aktivnosti, je lahko sestavljena tudi iz različnih vejitev, zank, prekinitev ali poslušalcev, a je znana vnaprej. Zaporedni delovni tokovi so v WF predstavljeni kot osnovni tip delovnih tokov, ki so v glavnem namenjeni razreševanju strukturiranih problemov, kjer je znan postopek obravnave problema, nepričakovanih dogodkov pa v večini primerov ni (Božnik, 2010, str. 17; White, 2013, str. 35);

- avtomat stanj (angl. *state-machine*): bistvo delovnih tokov, ki temeljijo na avtomatu stanj, so stanja sama. To predstavlja stanje vseh spremenljivk poslovnega procesa v točno določenem trenutku. Pojem "stanje", ki ga imamo v mislih, lahko ponazorimo s primerom klimatske naprave, ki lahko ohlaja, lahko greje, lahko je v pripravljenosti ipd. Vsaka od naštetih funkcionalnosti predstavlja stanje, v katerem je klimatska naprava, npr. v stanju hlajenja. Pomembno je, da je lahko hkrati le v enem izmed stanj (Božnik, 2010, str. 19; White, 2013, str. 29);

- diagram toka podatkov (angl. *flowchart*): omogoča modeliranje procesnega toka podatkov, ki ni predvidljiv, niti ne potrebuje zunanjih vplivov, ki bi bili vezani izključno na odločitve človeškega uporabnika poslovne programske rešitve, osnovne značilnosti delovnih tokov, ki jih razvijalci v WF upodabljajo v obliki diagrama toka podatkov so (White, 2013, str. 27):

- transparentne odločitvene možnosti;
- naravni tok lahko vodi tudi nazaj do predhodno izvedenih aktivnosti;
- primeren za odločitvene delovne tokove.

3.1.2.1.4 ORODJE ZA RAZVOJ DELOVNIH TOKOV

Pri orodju za razvoj delovnih tokov gre za oblikovalsko okolje, kjer razvijalec programske rešitve z uporabo tehnologije WF pripravi model poslovnih procesov v zapisu, ki ga podpira WF. Trenutna različica Windows Workflow Designer okolja je prisotna v programski opremi Visual Studio 2015. Gre za okolje, kjer je mogoča grafična obdelava in razhroščevanje (angl. *debugging*) modelov poslovnih procesov in posledično poslovnih programskih rešitev, ki temeljijo na modelih delovnih tokov v oblikah, ki jih podpira tehnologija WF (Developing Applications with the Workflow Designer, 2016).

3.1.2.1.5 PODATKOVNI MODEL DELOVNIH TOKOV

Delovni tokovi za procesiranje informacij potrebujejo podatke, ki jih lahko v WF pridobivajo na tri različne načine, podatkovni model pa je prikazan v Tabeli 1:

Tabela 1: Podatkovni model WF

PODATKOVNI MODEL	**OPIS MODELA**
Spremenljivke (angl. *Variable*)	Vrednosti spremenljivk so določene ob zagonu delovnega toka oz. programske rešitve ter shranjene kot del stanja instance delovnega toka.
Parametri (angl. *Argument*)	Nadzirajo pretok podatkov s sprejemanjem in pošiljanjem podatkov v ter iz same aktivnosti. Mogoče smeri parametrov so *In, Out* ter *InOut.*
Izrazi (angl. *Expression*)	Aktivnost z visoko stopnjo vrnjenih vrednosti, ki se uporabljajo pri povezovanju parametrov.

Vir: B. White, Pro WF 4.5, 2013, str. 51.

3.1.2.1.6 STORITVE ZA OHRANJANJE STANJA

Ohranjanje stanja (angl. *persistence*) pri WF na nek način predstavlja stanje pripravljenosti delovnega toka. Cilj tega je enak kot pri ostali sistemih in napravah, torej varčevanje z resursi, to pomeni z energijo, delovnim spominom, procesorsko močjo. Izvajalnik delovnega toka WF med izvajanjem programske rešitve zazna, da je programska rešitev ali del te (delovni tok) postal nedejaven in ga zato pretvori v podatkovni format, ki ga je mogoče zapisati v podatkovno bazo SQL. To običajno velja, ko se izvajajo procesi z dolgim časom izvajanja (Persistence Overview, 2016).

3.1.2.1.7 NADZOR DELOVANJA DELOVNIH TOKOV

Eden izmed večjih izzivov razvijalcev poslovnih programskih rešitev in programskih rešitev nasploh je razumevanje tistega, kar se dogaja v ozadju izvajajočih se aplikacij, vse to z namenom učenja in izboljšanja programske opreme. Pri gradnji poslovnih programskih rešitev, ki temeljijo na delovnem toku, je to še nekoliko bolj pomembno, saj je potrebno primerjati izvajanje samodejno generirane programske kode z modelom poslovnega procesa za ugotavljanje morebitnih neskladij. WF v osnovi podpira tovrstno nadzorovanje izvajanja delovnih tokov ter podatke zapisuje v Event Tracking for Windows (ETW), ki pa je že del operacijskih sistemov družine Microsoft Windows (White, 2013, str. 57).

3.1.2.1.8 SERIALIZACIJA

Serializacija je postopek, s pomočjo katerega je mogoče shraniti notranje stanje objektov v obliko, ki je primerna za shranjevanje ali prenos. Cilj serializacije je ohranjanje stanja objekta z namenom kasnejše uporabe ali uporabe v drugem sistemu. Tehnologija WF omogoča serializacijo in deserializacijo delovnih pravil, aktivnosti in tokov. Tako je omogočeno hranjenje njihovega stanja, mogoče jih je uporabiti v markup-datotekah, prikazovati je mogoče njihove lastnosti, polja ter dogodke v aplikaciji za oblikovanje delovnih tokov (Workflow Designer). WF omogoča privzete serializacijske zmožnosti za osnovne, že vgrajene aktivnosti, mogoče pa je narediti serializator po meri za tiste aktivnosti, ki niso del osnovne sheme WF. S serializatorjem po meri je mogoče določiti, kaj naj se serializira ter kako (Serialization Overview, 2016).

3.1.2.1.9 OZNAČEVALNI JEZIK

Windows Workflow Foundation razvijalcem omogoča deklarativni način razvoja delovnih tokov s pomočjo označevalnega jezika XAML (angl. *eXtensible Application Markup Language*), ki ga v primeru uporabe skupno s tehnologijo WF Microsoft definira kot *workflow markup*. Datoteke tega tipa je mogoče enostavno prevesti v delovni tok, jih naložiti v izvajalno okolje delovnih tokov, razvojno ogrodje pa omogoča tudi prevod v tip delovnega toka s t. i. datotekami *code-behind*, ki so implementirane v C# ali Visual Basic.

XAML, izvira iz jezika XML, je uporabljen v veliko komponentah ogrodja .NET, v WF pa predstavlja alternativno orodje za razvoj aktivnosti po meri in posledično delovnih tokov. Tako kot ostali programski jeziki, ki jih podpira ogrodje .NET, se tudi XAML pred izvajanjem prevede v CLR-kodo, ki omogoča uporabo več programskih jezikov hkrati v eni programski rešitvi (Workflow Markup Overview, 2016).

3.2 Java in Business Process Execution Language

3.2.1 Java

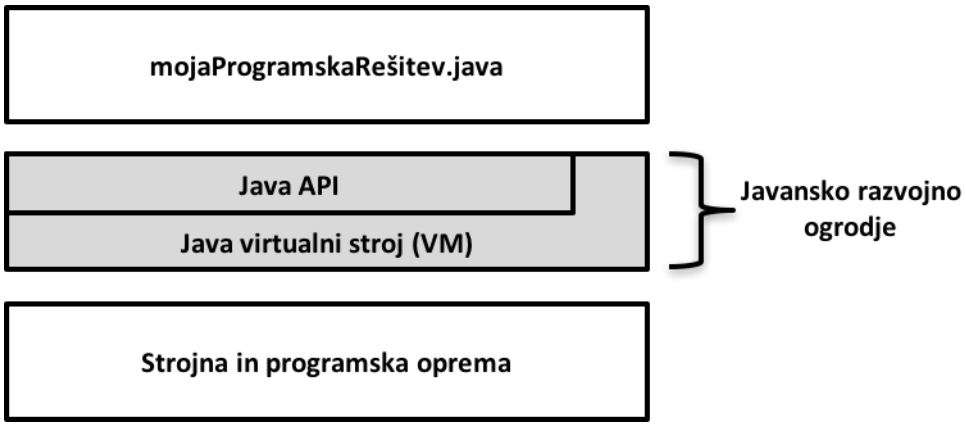
Javanska platforma obstaja trenutno v treh različicah, te so (Kurbel, 2008, str. 123):

- Java ME (Micro edition);
- Java SE (Standard edition);
- Java EE (Enterprise edition).

Pred letom 2006 so bile omenjene različice poznane pod imeni J2ME, J2SE in J2EE, kjer je »J2« pomenilo Java 2 platforma. Različica Java ME je namenjena uporabi na mobilnih napravah, kot so prenosni telefoni, tablični računalniki in podobno, Java SE vsebuje osnovne komponente za razvoj, implementacijo in izvajanje spletnih in lokalnih informacijskih sistemov v javanskem okolju, Java EE pa je dodatek k različici Java SE, ki omogoča tudi implementacijo t.i. porazdeljenega računalništva (angl. *distributed computing*) in večslojnih poslovnih programskih rešitev (Kurbel, 2008, str. 124).

Javanska platforma določa, kako bodo javanske programske rešitve razvite in izvedene. V arhitekturni shemi se to razvojno ogrodje nahaja med javansko programsko rešitvijo (rezultatom razvoja) in spodaj ležečo platformo, ki je sestavljena iz strojne opreme in operacijskega sistema. Ključna sestavna dela javanskega razvojnega ogrodja sta Java virtual machine (javanski virtualni stroj – Java VM) ter Java application programming interface (Java API), kot prikazuje tudi Slika 13 (Kurbel, 2008, str. 125).

Slika 13: Funkcija Javanske platforme

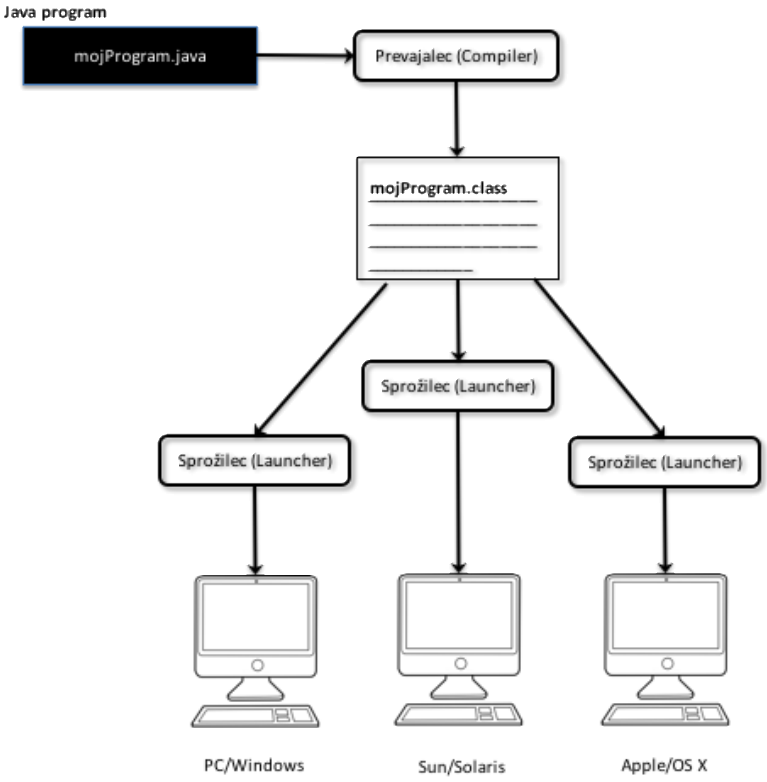


Vir: Prirejeno po M. Campione, K. Walrath & A. Huml, The Java Tutorial – A Short Cource on the Basics (3rd ed.), 2001, str. 4.

Javanski virtualni stroj je v osnovi virtualno računalniško okolje, ki lahko prevede in izvaja javanske programske rešitve. Javanski API je sestavljen iz velike zbirke že razvitih programskih

gradnikov, programskih komponent, ki so porazdeljene v knjižnice in gradniške pakete na podlagi ustreznih gradniških razredov in vmesnikov. Primer takšnega gradniškega paketa je paket za razvoj grafičnega uporabniškega vmesnika (Kurbel, 2008, str. 125).

Slika 14: Prevajanje in izvajanje javanskih programskih rešitev



Vir: Prirejeno po M. Campione et al., The Java Tutorial – A Short Cource on the Basics (3rd ed.), 2001, str. 4.

Javanska platforma je neodvisna od strojne opreme in operacijskega sistema, delovanje je omogočeno na več kombinacijah operacijskih sistemov in strojne opreme. Takšna neodvisnost je dosežena z uporabo dvonivojskega procesiranja, v katerem so javanski programi dvakrat prevedeni. Na Sliki 14, kjer so prikazani omenjeni nivoji, je razviden omenjeni koncept delovanja (Kurbel, 2008, str. 125):

- prevajalec (angl. *compiler*) prevede javansko programsko kodo v posredniški jezik, ki se imenuje *Java Bytecode* in je neodvisen od konfiguracije strojne in programske opreme;

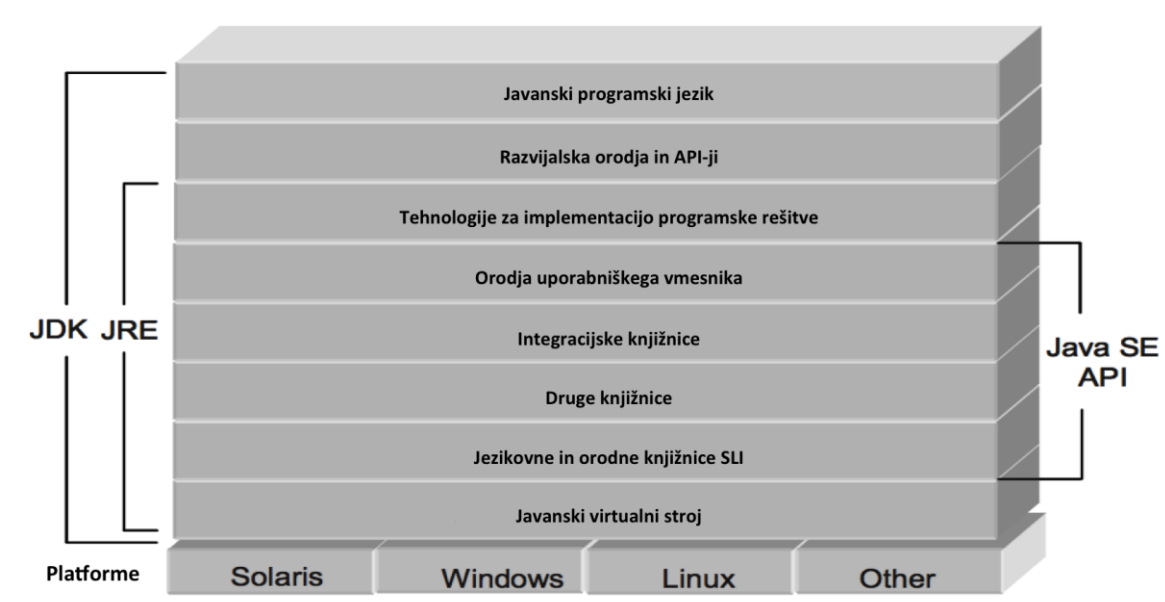
- sprožilec (angl. *launcher*) v sklopu Java VM (javanskega virtualnega stroja), preveden v obliko, ki jo podpira posamezni javanski virtualni stroj, ki je nameščen in se izvaja okviru posameznih konfiguracij strojne in programske opreme (Windows, OS X, Solaris). Vsaka različica javanskega virtualnega stroja lahko kot vhodni podatek dobi kakršno koli Java Bytecode ter kodo nato prilagodi platformi in izvede programsko rešitev. Posebnost Java Bytecode je v tem, da je s strani sprožilca, ki deluje v sklopu javanskega virtualnega stroja, lahko prevedena ali interpretirana, hkrati pa sam javanski virtualni stroj po prevodu Java

Bytecode v naravni (java) programski jezik programsko kodo dodatno optimizira na podlagi zagonskih lastnosti programske rešitve.

3.2.1.1 JAVA SE

Omenili smo že, da sta različici Java SE in Java EE med seboj povezani, ter da je Java EE nadgradnja, ki je namenjena kompleksnejšim poslovnim programskim rešitvam, zato bomo najprej predstavili različico SE, ki je ključna za razumevanje različice EE.

Slika 15: Zgradba javanskega ogrodja



Vir: E. K. Kurbel, The Making of Information Systems, 2008, str. 126.

Standardna verzija ogrodja Java zajema večje število razvijalskih vmesnikov, ki služijo za razvoj, vgradnjo (implementacijo) in izvajanje poslovnih programskih javanskih rešitev. Uradna dokumentacija proizvajalca skupek tehnologij in razvijalskih orodij označuje pod imenom Java razvijalski kit (angl. *Java Development Kit* – JDK). Skupek razvojnih orodij pod imenom JDK predstavlja javansko razvojno ogrodje. Na Sliki 15 je predstavljena zgradba ogrodja oz. JDK, hkrati pa so predstavljeni tudi posamezni gradniki, ki predstavljajo posamezne ključne elemente javanskega ogrodja (Kurbel, 2008, str. 126):

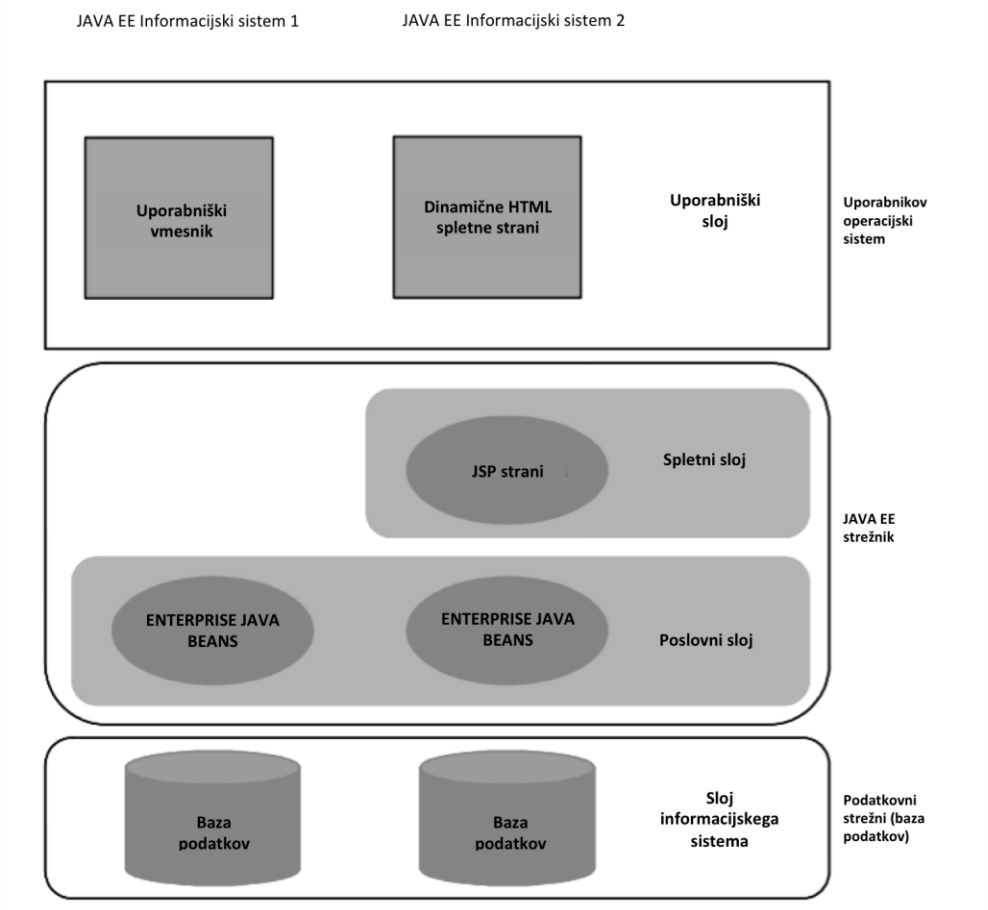
- Java programski jezik;
- razvojna orodja za gradnjo programskih rešitev in API (prevajalec, zagonski element ...),
- tehnologije za implementacijo programske rešitve;
- orodja uporabniškega vmesnika;
- integracijske in druge knjižnice;
- jezikovne in orodne knjižnice;

- javanski virtualni stroj.

3.2.1.2 JAVA EE

Veliko število večjih poslovnih informacijskih sistemov v današnjem času deluje po principu porazdeljenih sistemov z večplastno arhitekturo, kar omogoča dostop in procesiranje transakcij velikega števila uporabnikov ob istem času. To je potreba, ki jo imajo organizacije pri informatizaciji poslovanja. Javanska platforma podpira razvoj in implementacijo takšnih poslovnih programskih rešitev, vendar je za to potrebna uporaba različice Java EE. Java EE zajema vse komponente, kot so prikazane na Sliki 15, hkrati pa podpira še dodatne funkcije in vsebuje komponente, ki omogočajo gradnjo in razvoj distribuiranih informacijskih sistemov. Običajno so informacijski sistemi, ki so grajeni z uporabo Jave EE, sestavljeni iz štirih slojev, kot je prikazano na Sliki 15 (Kurbel, 2008. str. 127).

Slika 16: Primer dveh arhitektur v Java EE



Vir: E. K. Kurbel, The Making of Information Systems, 2008, str. 127.

Logična arhitektura je pogosto v fizični obliki prikazana kot trislojna, ker se običajno ogrodje Java EE nahaja na strežniku (Java EE strežnik), uporabniški vmesnik (klient) pa deluje na uporabniškem računalniku, vendar gre v svoji naravi za štirislojno arhitekturo. Prav tako se sloj

EIS (Enterprise Information Systems), ki zajema baze podatkov ter t. i. podedovane informacijske sisteme (angl. *legacy systems*), izvaja na svojem strežniškem sistemu. Temu je običajno tako, ker se v sklopu informacijskega sistema izvajajo tudi podedovani informacijski sistemi, ki ne temeljijo na javanskem ogrodju in ne uporabljajo javanskih tehnologij (Jendrock, Cervera - Navarro, Evans, Gollapudi, Haase, Markito & Srivathsa, 2013).

Kot je razvidno s Slike 16, obstajata dve vrsti informacijskih sistemov v ogrodju Java EE; informacijski sistem 2 je spletni informacijski sistem, kjer uporabnik interagira s pomočjo spletnega brskalnika na uporabniškem računalniku, z uporabo tehnologije HTML in JSP (JavaServer Pages), kjer se generira dinamična vsebina spletnih strani. Uporabnik informacijskega sistema 1 pa v nasprotju s sistemom 2 uporablja grafični uporabniški vmesnik, ki je bil razvit z uporabo javanskih tehnologij, kot je Swing, AWT in podobnih (Kurbel, 2008. str. 128).

Temeljna razlika med standardno (SE) in napredno, poslovno, (EE) različico platforme Java je v tem, da morajo komponente Jave EE spoštovati specifična pravila in omejitve, ki se pojavljajo pri izvajanju in razvoju poslovnih programskih rešitev (Jendroch et al., 2013).

3.2.2 Business Process Execution Language

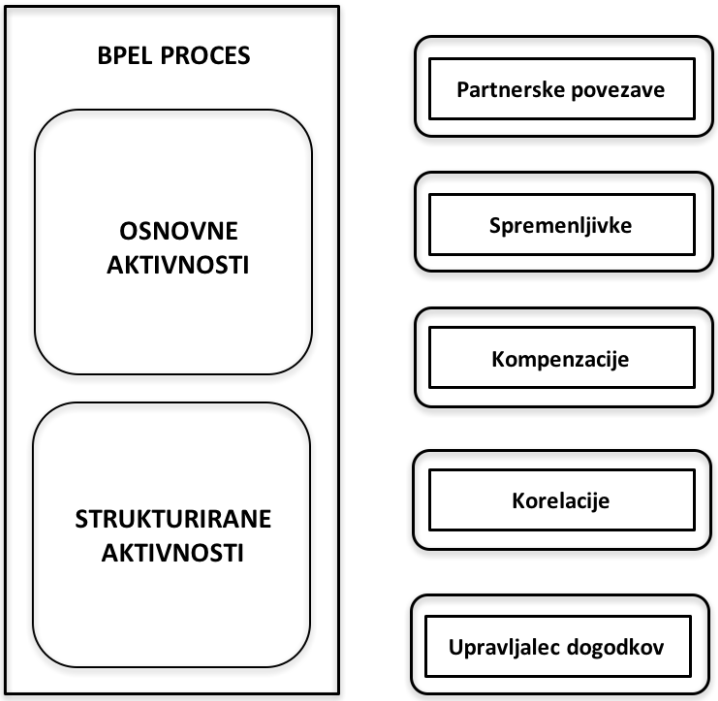
Poleg tehnologije Windows Workflow Foundation je temelj raziskave magistrske naloge tudi tehnologija BPEL. To je programski jezik, ki omogoča definiranje in izvajanje delovnega toka v povezavi s spletnimi storitvami. Ta omogoča kompozicijo (pripravo sestavljenih spletnih storitev iz že obstoječih spletnih storitev) in orkestracijo spletnih storitev, obenem pa predstavlja najbolj razširjen in specializiran jezik za avtomatizacijo poslovnih procesov. Gre za naslednika jezika XLANG, ki je bil razvit pri podjetju Microsoft, in jezika WSFL, ki ga je razvilo podjetje IBM. Prvo različico jezika pod imenom BPEL4WS je leta 2003 standardizirala agencija za standardizacijo OASIS (Oasis, 2007, str. 6). Poslovne programske rešitve, ki temeljijo na BPEL, je nujno razvijati skupaj s standardi WSDL, SOAP in podobnimi tehnologijami, ki omogočajo spletne storitve.

BPEL tehnologija je že ob razvoju bila usmerjena predvsem k podpori spletnim storitvam in paradigmi SOA. V arhitekturi informacijskega sistema ima BPEL mesto na nivoju tehnologij spletnih storitev. Glavna logika delovanja tehnologije temelji na tem, da programski jezik oblikuje upravljalno semantiko interakcij med spletnimi storitvami, ki so predmet razvoja. Za komunikacijo med posameznimi aktivnostmi procesa in med posameznimi poslovnimi procesi se uporablja, tako kot pri tehnologiji Workflow Foundation, opisni programski jezik WSDL. Bistvo obravnavane tehnologije in sama logika, na kateri temelji, je obravnavanje posameznega delovnega toka kot spletne storitve. Cilj tega je boljša komunikacija in interakcija znotraj delovnih tokov, enostavnejša ponovna uporaba poslovnih procesov in boljše možnosti nadgradnje informacijskih sistemov (Khalaf , 2005, str. 317–342).

3.2.2.1 OSNOVNI KONCEPTI

Procesi, predstavljeni s tehnologijo BPEL, so lahko izvajalni ali abstraktni. Izvajalni procesi so tisti, ki jih lahko izvajalnik delovnega toka dejansko izvaja, medtem ko so abstraktni procesi definicija protokola ali pa javno dostopen opis obnašanja posameznega partnerja. Iz tega izhaja, da je vloga abstraktnih procesov informativne narave, služi lahko kot vzorec za različne procese. Programska koda se med izvajalnim in abstraktnim procesom razlikuje v tem, da se v izvajalnem poslovnem procesu razvija celoten proces, v abstraktnem pa se modelira le aktivnosti, ki sodelujejo v posameznem delovnem toku ter tiste, ki javno komunicirajo z ostalimi partnerji (Oasis, 2007, str. 147–163).

Slika 17: Struktura BPEL



Vir: O. Coupelon, Analyse et optimisation de l'architecture des moeurs BPEL, 2007, str. 13.

V osnovi jezik BPEL predstavlja jezik za orkestracijo. Koncept orkestracije definira izvršljiv proces, ki vključuje izmenjavo sporočil z zunanjimi sistemi, kjer izmenjavo sporočil nadzoruje oblikovalec orkestracije. Potek orkestracije vodi tako imenovani dirigent, nameščen v osrednjem nadzornem centru. Ta skrbi za način vodenja celotnega porazdeljenega sistema, sestavljenega iz več različnih elementov. Poslovni procesi jezika BPEL so opisani na podlagi definicije jezika v obliki XML standarda (Oasis, 2007, str. 147–163). Predstavitev poslovnega procesa, zapisanega na podlagi notacije XML, je možna tudi v grafični obliki, a ta ni standardizirana. S pomočjo notacije XML je mogoče definirati omejen nabor konstruktov, ki predstavljajo aktivnosti poslovnega procesa. Izvajanje poslovnih procesov poteka v posebnem okolju na ločenem procesnem strežniku. Končni uporabnik do njih dostopa preko spletnih storitev, kar pomeni, da so delovni tokovi obravnavani kot navadne spletne storitve. Dostop do teh s strani delovnih

tokov je zagotovljen na podlagi protokola SOAP in operacij spletnih storitev, definiranih v obliki datoteke WSDL (Web Service Definition Language) (Ribić, 2015, str. 6).

Ključne zmogljivosti BPEL tehnologije so (Khalaf , 2005, str. 317–342):

- paralelno izvajanje aktivnosti;
- koordiniranje asinhronih komunikacij med spletnimi storitvami;
- povezana izmenjava sporočil med vpletenimi strankami;
- manipulacija s podatki pri iteracijah med partnerji;
- podpora za poslovne transakcije in aktivnosti, ki se izvajajo dlje časa (angl. *long running transactions*);
- podpora za konsistentno ravnanje z napakami.

Osnovne komponente BPEL tehnologije (Khalaf , 2005, str. 317–342; Coupelon, 2007, str. 13–16), kot so prikazane na Sliki 17, so podrobneje predstavljene v sledečih podpoglavjih.

3.2.2.1.1 PARTNERJI IN PARTNERSKE POVEZAVE (angl. *Partner Links*)

V vsakem poslovnem procesu je potrebno definirati nabor specifikacij partnerskih povezav, ki opredeljujejo zahtevane funkcionalnosti v razmerju do vsakega izmed partnerjev, deležnikov, procesa.

3.2.2.1.2 AKTIVNOSTI (angl. *Activities*)

Aktivnosti v tehnologiji BPEL imajo prednastavljene načine obnašanja ter so razdeljene v dve temeljni kategoriji:

- STRUKTURIRANE oz. SESTAVLJENE AKTIVNOSTI (angl. *Structured Activities*): ta tip aktivnosti narekuje vedenjske in izvajalne smernice v delovnem toku, specificirajo namreč posamezne korake poslovnega procesa. Primeri funkcij strukturiranih aktivnosti so (Ribić, 2015, str. 8):

    - zaporedje izvajanja aktivnosti;
    - tokovi različnih aktivnosti, ki se lahko izvajajo vzporedno;
    - pogojno izvajanje aktivnosti;
    - zanke in izbira alternativnih poti;
    - povezovanje z zunanjimi partnerji;
    - uvoz podatkov iz zunanjih dokumentov;
    - konstrukti, ki skrbijo za definiranje spremenljivk za izvajanje delovnih tokov;

- PRIMITIVNE oz. OSNOVNE AKTIVNOSTI (angl. *Basic Activities*) (Khalaf, 2005, str. 317–342): nahajajo se na skrajnih delih procesa, njihova naloga pa je izvajanje osnovnih

operacij, kot so pošiljanje/sprejemanje sporočil, klic novih storitev, čakanje in opozarjanje na napake.

3.2.2.1.3 MANIPULACIJA PODATKOV (angl. *Data Manipulation*)

Poslovni procesi, ki jih upravlja BPEL tehnologija, vsebujejo stanje (angl. *stateful processes*), stanje pa vsebuje sledeče vsebine (Khalaf , 2005, str. 317–342; Coupelon, 2007, str. 13–16):

- hranjenje aplikacijskih podatkov, ki jih porabljajo zaledne programske rešitve, ki jih upravlja posamezni delovni tok;

- hranjenje kontekstualnih podatkov, ki imajo posebno vrednost za srednji sloj v arhitekturi poslovnih programskih rešitev in niso vidni ostalim programskim rešitvam, ki jih upravlja poslovni proces, kot npr. ID-transakcije in žetoni posamezne instance procesa.

Prikaz podatkov je sicer mogoč le z uporabo drugih standardov in konceptov, saj je tehnologija BPEL uporabna samo z uporabo še drugih, dopolnilnih tehnologij. Gre namreč za tehnologijo, ki temelji na XML-prikazu podatkov.

3.2.2.1.4 KOMPENZACIJA (angl. *Compensation*)

Kompenzacija je ena kritičnih operacij BPEL-tehnologije. Smisel kompenzacije je v načinu zaustavitve in ponovne vzpostavitve transakcij. Ob zaustavitvi določene transakcije in ponovni vzpostavitvi so se lahko zgodile spremembe, ki niso več zaželene. Kompenzacija omogoča neke vrste filtracijo neželenih, a že opravljenih sprememb (Khalaf, 2005, str. 317–342; Coupelon, 2007, str. 13–16).

3.2.2.1.5 KORELACIJA (angl. *Correlation*)

Gre za proces, ki omogoča asociacijo vhodnih sporočil s pravilno instanco delovnega toka. To je urejeno s pomočjo dodelitve enkratne identifikacijske številke, ki omogoča dostavo sporočila pravi instanci (Khalaf , 2005, str. 317–342; Coupelon, 2007, str. 13–16).

3.2.2.1.6 UPRAVLJAVEC DOGODKOV (angl. *Event handler*)

Vsak poslovni proces lahko vsebuje upravljavca dogodkov. Ob specifičnem dogodku se delovni tok premakne na aktivnost, ki upravlja tovrstne specifične dogodke. Upravljavci dogodkov se razlikujejo od običajnih in strukturiranih aktivnosti v tem, da se izvršijo samo takrat, ko se zgodi določen dogodek. Obstajata dve vrsti dogodkov, na katere BPEL tehnologija reagira na specifičen način (Khalaf , 2005, str. 317–342; Coupelon, 2007, str. 13–16):

- sporočilni dogodki (angl. *Message Events*): dospetje specifičnega sporočila;
- časovni dogodki (angl. *Alarm Events*): čakanje določeno število ur, minut, sekund ipd.

3.2.2.2 IZVAJALNIKI BPEL

Tehnologija BPEL v primerjavi z Workflow Foundation nastopa samostojno, v obliki standarda, brez pripadajočega izvajalnika delovnih tokov. Ta je za delovanje seveda potreben, vendar ga mora vsak izmed ponudnikov razvojnih orodij, ki želi implementacijo tehnologije BPEL v svoje orodje, razviti sam. Izvajalnik je tisti del tehnologije BPEL, ki je sposoben interpretacije programske kode. Nekateri ponudniki (komercialni kot tudi odprtokodni) izvajalnikov BPEL-tehnologije so predstavljeni v Tabeli 2 (Khalaf , 2005, str. 317–342; Coupelon, 2007, str. 13–16).

Tabela 2: Nekateri BPEL-izvajalniki

IZVAJALNIK	BPEL RAZLIČICA	VRSTA LICENCE
ActiveBPEL Engine	1.1 in 2.0	odprtokodna
Apache ODE	2.0	odprtokodna
IBM WebSphere	2.0	komercialna
SAP Exchange Infrastructure	2.0	komercialna
ORACLE BPEL Process Manager	2.0	komercialna

Vir: L. Sikos, Mastering Structured Data on the Semantic Web, 2015, str. 141.

4 ANALIZA SMERI RAZVOJA IN PRIMERJAVA TEHNOLOGIJ

4.1 Analiza smeri razvoja tehnologij

V predhodnih razdelkih smo opisali koncepte in tehnologije, ki jih bomo v tem poglavju raziskali. Podlaga, zakaj smo se podrobneje osredotočili na tehnologiji Business Process Execution Language (skrajšano BPEL) in Windows Workflow Foundation (skrajšano WF) pa sledi v nadaljevanju tega poglavja.

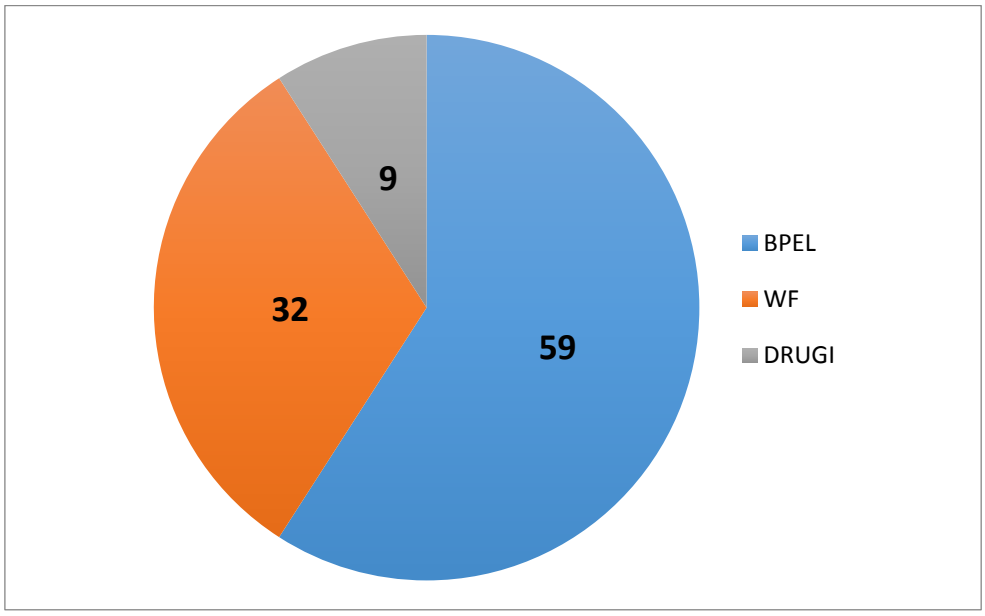
4.1.1 Stanje na področju uporabe tehnologij

Gartner, svetovno znano podjetje, ki opravlja raziskave in svetovalno dejavnost na področju informacijskih tehnologij, vsakoletno objavlja raziskave na več področjih. Eno izmed teh področij so tudi orodja za razvoj poslovnih programskih rešitev, ki temeljijo na poslovnih procesih. Gartner za omenjena orodja uporablja kratico BPMS, ki predstavlja angleški izraz Business Process Management Suites. V nalogi se sicer ne osredotočamo na razvojna orodja, temveč na jedrne tehnologije, ki so ključne za razvoj poslovnih programskih rešitev, ki temeljijo na poslovnih procesih. Na podlagi Gartnerjevih (Hill, Cantara, Kerremans & Plummer, 2009, str. 1–34) raziskav je razvidno, da orodja temeljijo na tehnologiji BPEL v povezavi z javansko platformo oz. na tehnologiji Windows Workflow Foundation v povezavi z Microsoftovo platformo .NET (Hill et al., 2009, str. 4–5). V dotični raziskavi sicer ni izrecno poudarjeno, da je BPEL-tehnologija, na kateri dotična orodja temeljijo, dejstvo pa je, da je BPEL t. i. *de-facto*

industrijski standard na področju izvajanja delovnih tokov, kot to trdijo Ouyang, van der Aalst, Dumas in ter Hofstede (2006, str. 2), Jurišič (2011, str. 163–171) in Jurič in Pant (2008, str. 1), sam BPEL pa deluje v kombinaciji z javansko platformo. Iz Gartnerjeve raziskave izhaja, da so orodja, ki so najmnožičneje zastopana na tržišču, dveh vrst, tista, ki temeljijo na Microsoftovem .NET razvojnem ogrodju, in tista, ki temeljijo na javanskem razvojnem ogrodju. Na podlagi tega torej posledično izhaja tudi dejstvo, da sta prevladujoči jedrni tehnologiji za razvoj poslovnih programskih rešitev, ki temeljijo na poslovnih procesih, Windows Workflow Foundation in Business Process Execution Language.

Za primerjavo smo kot prvo izbrali raziskavo, ki temelji na letu 2008, kjer so razvidni prvi podatki od uvedbe tehnologije Workflow Foundation, ki je bila predstavljena leta 2006. Leta 2008 je Gartner na podlagi pregleda tržišča ugotovil, da lahko v svojo raziskavo uvrsti 22 različnih proizvajalcev orodij za razvoj poslovnih programskih rešitev, ki temeljijo na poslovnih procesih (Hill et al., 2009, str. 3), od tega jih 13 (59 odstotkov) temelji tudi na tehnologiji BPEL, 7 (32 odstotkov) tudi na tehnologiji Workflow Foundation, le 2 (9 odstotkov) proizvajalca pa uporabljata druge tehnologije (Slika 18).

Slika 18: Razmerja uporabljenih tehnologij na podlagi Gartnerjeve raziskave iz leta 2008 v %

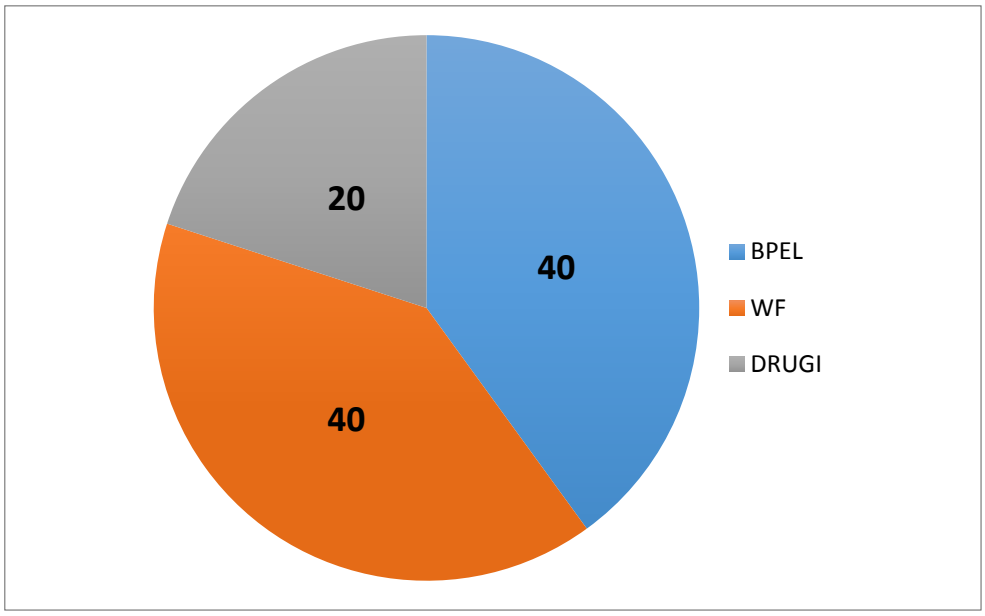


Vir: Prirejeno po J. B. Hill et al., Magic Quadrant for Business Process Management Suites, 2009, str. 3.

Kot primerljivo raziskavo smo izbrali ravno tako Gartnerjevo (Duine, Shulte, Cantara, & Kerremans, 2015, str. 1–31), ki temelji na letu 2014, ta pa je v času izdelave te magistrske naloge najbolj ažurna. Gartner je v tem času sicer nekoliko spremenil kriterije analize, saj se tehnologija in zahteve, povezane s poslovnimi programskimi rešitvami, spreminjajo, sicer pa je to dejstvo, ki zadeva samo logiko delovanja orodij, ki se prilagajajo potrebam razvijalcev in uporabnikom. Za potrebe magistrske naloge so podatki pridobljeni po novi metodi dobrodošli, saj nas zanima, kakšno je razmerje uporabljenih tehnologij v zadnjem času. Od 15 orodij, ki so bila uvrščena v

Garnerjevo raziskavo (AgilePoint Positioned in Gartner's Magic Quadrant for Intelligent Business Process Management Suites 2015 Report), jih 6 (40 odstotkov) temelji na javanski platformi in tudi na tehnologiji BPEL, 6 (40 odstotkov) jih temelji na ogrodju .NET in tudi na tehnologiji Workflow Foundation, 3 orodja (20 odstotkov) pa temeljijo na drugih tehnologijah (Slika 19). Kako je prišlo do takšnega razvoja tehnologij in takšnih razmerij, opisujemo v naslednjem razdelku tega poglavja.

Slika 19: Razmerja uporabljenih tehnologij na podlagi Gartnerjeve raziskave iz leta 2014 v %



Vir: Prirejeno po R. Duine et al., Magic Quadrant for Intelligent Business Process Management Suites, 2015, str. 6.

4.1.2 Ozadje razvoja tehnologij

V sredini 90. let prejšnjega stoletja, ob pojavu množične uporabe interneta, so se začele razne eksperimentalne študije novih poslovnih modelov in poizkusov integracije B2B poslovnih modelov preko spletnih tehnologij. Zaradi poplave poizkusov so se pojavili tudi prvi predlogi za standardizacijo protokolov in tehnologij sporočanja in izmenjave dokumentov med poslovnimi partnerji, kar je pripeljalo do množične uporabe tehnologije XML za poslovno komunikacijo. Sama izmenjava sporočil in dokumentov pa ni bistveno prispevala k lažjemu upravljanju celotnega scenarija elektronskega poslovanja, težava je bila v pomanjkanju "orkestracije". Tovrstno razmišljanje je pripeljalo do spoznanja, da je potrebno informatizacijo poslovanja tesno povezati z managementom poslovnih procesov. Z razvojem dogodkov v tej smeri se je v začetku leta 2000 pojavil prvi odgovor na pomanjkanje skupnega programskega jezika, ki bi bil sposoben prikaza, izvedbe in orkestracije delovnih tokov, razvila ga je organizacija "Business Process Management Initiative" – BPMI, to sta bila programski jezik BPML in notacija BPMN. Šlo je za radikalno inovacijo na področju izvajanja delovnih tokov in razvoja poslovnih programskih rešitev, temeljila pa sta na konceptu t. i. modelno vodene arhitekture (angl. *Model-driven Architecture*). Notacija BPMN predstavlja grafično upodobitev programskega jezika BPML, v

katerem so gradniki procesa prikazani kot podatki, ki jih nato izvaja zunanji izvajalnik programskega jezika BPML. Cilj razvijalcev tehnologije je bila množična uporaba jezika, standardizacija, ki bi na področju razvoja poslovnih programskih rešitev predstavljala neke vrste revolucijo.

Razvoj rešitve po meri uporabnikov bi z uporabo tovrstnih tehnologij namreč bil mogoč na enostavnejši in hitrejši način, hitrejše pa bi bilo tudi nadaljnjo prilagajanje spremembam v poslovnemu modelu, ki bi se odrazile na procesih in aktivnostih, mogoča pa bi bila tudi ponovna uporaba razvitih rešitev ali delov le-teh (Smith & Fingar, 2003, str. 4–15).

4.1.2.1 Razvoj tehnologije Business Process Execution Language

Kljub navdušenju strokovne javnosti nad razvitimi tehnologijami, ki jih je razvila organizacija BPMI, je bil uspeh le delni. Vodilnih podjetij na trgu orodij za razvoj poslovnih programskih rešitev uporaba tehnologije BPML ni prepričala, vseeno pa so spoznali, da je to potrebna smer razvoja novih tehnologij, kar je pripeljalo do razvoja tehnologije BPEL. Smith, avtor članka »BPMS 2008 – Look Back to Look Forward« (Smith, 2008, str. 5–8), ki je izšel v sklopu revije organizacije BPTrends.com, ugotavlja, da je razvoj tehnologije BPEL splet nekaterih odločitev vodilnih organizacij na področju razvoja tehnologij, ki so povezane s poslovnimi procesi.

V devetdesetih letih 20. stoletja je tematika managementa poslovnih procesov doživela razcvet, istočasno se je računalniška podpora poslovanjem podjetij zelo razširila in postala dostopnejša, s tem pa tudi dostopnost programskih rešitev in tehnologij, informacijskih sistemov, ki so podpirali management poslovnih procesov, tovrstna programska oprema se je imenovala BPMS (Business Process Management Systems). Z nadaljnjim razvojem informacijskih tehnologij se je pojavila potreba po izvajanju delovnih tokov in razvoju programskih rešitev, ki bi neposredno temeljile na poslovnem modelu. Neprofitna organizacija BPMI.org je tako leta 2000 razvila prvi programski jezik, odprtokodni BPML, ki je omogočal grafično oblikovanje poslovnih procesov, hkrati pa tudi generacijo izvršljive programske kode.

Panoga je novost z navdušenjem sprejela, kmalu pa so se vodilna podjetja na področju informacijske tehnologije želela vključiti v nadaljnji razvoj. Posebej nujno je to postalo ob pojavu tehnologije spletnih storitev (angl. *Web Services*). Večina razvijalcev poslovnih programskih rešitev namreč te gradi iz manjših modulov, delcev krovnega poslovnega procesa, ki se imenujejo podprocesi, vsak izmed njih pa je informacijsko podprt ter povezan s preostalimi delci, vsak zase je svoja spletna storitev in skupaj sestavljajo paradigmo SOA.

Ključna pobudnika za razvoj konkurenčne tehnologije obstoječemu jeziku BPML sta bila Microsoft in IBM. Vedela sta, da morata tehnologije spletnih storitev imeti pod nadzorom, ker gre za novosti, ki bodo krojile vsaj prihodnje desetletje informacijskih tehnologij. Vsak po svoje sta razvijala avtorski programski jezik, ki bi podpiral tovrstne tehnologije, zanimali pa so se za vse tehnologije, ki obstajajo v povezavi s SOA. Tako sta IBM in Microsoft hotela prevzeti tudi nadzor nad BPML, ker pa jima to ni uspelo, sta združila moči in ustvarila svoj standard. Nastal je

programski jezik BPEL (pri oblikovanju tehnologije so nato sodelovala še podjetja BEA Systems, SAP in Siebel Systems), v katerega sta podjetji vložili veliko sredstev, tudi za promocijo, ter tako tehnologijo BPML popolnoma zasenčili (Smith, 2008, str. 6–10).

Vendar pa Ko, Lee in Lee (2009, str. 744–791) poudarjajo, da je BPML v primerjavi z BPEL formalno popoln jezik, ki je sposoben prikaza in samostojnega izvajanja celotnega delovnega toka, v nasprotju z BPEL–jezikom, ki ni popoln programski jezik. Za izvajanje namreč potrebuje platformo, ki je običajno javanska, da zapolni semantične luknje, hkrati pa BPEL za izvajanje potrebuje tudi naknadno razvit BPEL-izvajalnik, BPML pa to je – programski jezik in izvajalnik. Iz tega izhaja dodatna spodbuda k temu, da so se razvijalci BPEL-tehnologije odločili za razvoj od začetka in ne za gradnjo na tehnologiji BPML. Šlo je namreč za kompleksen jezik, njegova celovitost pa je bila ovira. BPML namreč za razvoj poslovnih programskih rešitev in izvajanje le-teh ne potrebuje razvojnih ogrodij, temveč nastopa v celoti samostojno brez njega, ima namreč svoj izvajalnik in svoje razvojno okolje. Izvajanje poslovnih programskih rešitev, razvitih s tehnologijo BPML, ne poteka v sklopu razvojnega ogrodja. V nasprotju s tem pa platforme, kot sta Java in .NET, delujejo z nepopolnimi tehnologijami, ki za delovanje in izvajanje potrebujejo razvojno ogrodje s pripadajočimi komplementarnimi tehnologijami. Te skupaj tvorijo celoto poslovne programske rešitve. Takšna kombinacija omogoča enostavnejše manipuliranje in sodelovanje med različnimi tehnologijami.

Sama ovira oz. nezmožnost prevzeti nadzor nad tehnologijami torej najbrž ni bila ključni razlog, da sta se Microsoft in IBM odločila za razvoj BPEL-tehnologije in ne implementacije BPML, temveč je bila v ozadju odločitve ključna struktura izdelka. Na to nakazuje tudi dejstvo, da BPEL temelji na jeziku XML, vsi podatki, ki jih prenaša, pa so v obliki XML. Ključna pa je bila integracija nove tehnologije v obstoječa razvojna ogrodja ter seveda obvladovanje tehnologij spletnih storitev in vzpostavitev standarda. Sicer je BPEL tudi odprtokodna tehnologija, trenutno pod okriljem organizacije OASIS (Organization for the Advancement of Structured Information Standards), ki je samostojna, brez vgrajenega izvajalnika.

Sicer pa so se že v času snovanja BPEL-a ter takoj po uvedbi pojavile pomanjkljivosti tehnologije ter trenja med razvojnimi skupinami in idejami, ki so jih te imele glede končne podobe BPEL-tehnologije in vlog, ki jih bo ta imela pri razvoju novih poslovnih programskih rešitev, tako so posamezne skupine začele izdelovati in testirati predloge izboljšav oz. posebne različice tehnologije:

- BEA Systems in IBM, ključna člana razvojne skupine BPEL-tehnologije, sta predlagala razširitev razvojnih zmogljivosti BPEL v smislu dodajanja nekaterih zmogljivosti, ki bi BPEL-u omogočale »programerske sposobnosti«. BPEL je v osnovi namenjen programiranju večjih sistemov, ki so sestavljeni iz manjših modulov (aplikacij), skrbi za povezovanje med njimi ter krmili delovanje, orkestracijo. BEA in IBM sta želela dodati zmožnosti, ki bi procesnim definicijam dodale nekatere JAVA programerske izraze. Ti bi omogočali nekaj osnovnih operacij aktivnosti poslovnih procesov (npr. kalkulacije vrednosti, priprava dokumenta iz razpoložljivih informacij iz ostalih dokumentov, ipd.) in tako enostavnejši

razvoj celotne programske rešitve. Predlagan dodatek tehnologiji BPEL ima ime BPELJ, kjer črka J nakazuje na programski jezik JAVA (Blow, Goland, Kloppmann, Leymann, Pfau, Roller & Rowley, 2004, str. 3–24);

- nekateri ponudniki razvojnih orodij so BPEL-u dodajali dodatne XML-definicije, v enem primeru tudi več kot 50 novih oznak. To so počeli zaradi želje po diferenciaciji svoje BPEL-implementacije v primerjavi s konkurenti, s tem pa so hkrati nakazovali, da je tehnologija pomanjkljiva. Glavna kritika je bila, da BPEL ni sposoben učinkovito opravljati nalog, ki se jih zahteva od novega programskega jezika. Bil je bolj podoben dodatni funkciji v sklopu razvojnih orodij kot pa samemu programskemu jeziku oz. tehnologiji (Smith, 2005, str. 4);

- težava BPEL-tehnologije je, da mora vsak izmed ponudnikov razviti svoj BPEL-izvajalnik, kar se je v praksi odrazilo v potrebi po dodatnih vtičnikih manjših ponudnikov razvojnih orodij. Če so hoteli še naprej vzdrževati kompatibilnost njihovih orodij z ogrodji, ki jih nudijo vodilna podjetja, so morali svojo BPEL-rešitev in izvajalnik prilagoditi tako, da je razumel delovanje BPEL-izvajalnika razvojnih ogrodij vodilnih podjetij (Smith, 2005, str. 4);

- delovni tokovi običajno zahtevajo tudi človeško interakcijo, interakcijo z uporabniki poslovnega procesa, BPEL 1.1 in tudi BPEL 2.0 pa tega ne podpirata, omogočata samo avtomatizacijo poslovnih procesov in orkestracijo. Gre za eno večjih kritik tehnologiji. Kot meni Manoj Das, direktor oddelka za management poslovnih procesov pri podjetju Oracle, v intervjuju za revijo InfoQ (Little, 2008), ki ga v svojem gradivu omenja tudi organizacija OASIS: "BPEL v svoji naravi ne razlikuje med človeško in avtomatizirano aktivnostjo, kar pa je za večino uporabnikov postalo nujno potrebno. Razvoj BPEL4People prinaša ravno to, ločevanje med človeškimi in avtomatiziranimi aktivnostmi, med katerimi se značilnosti izvajanja bistveno razlikujejo". IBM in SAP sta zato naknadno razvila dodatek BPEL4People, ki omogoča ravno to, brez uporabe raznih dodatkov, ki so za osnovno različico WS-BPEL-tehnologije v arhitekturi nameščeni na višji plasti kot sama BPEL-tehnologija. Ključni koncepti, ki so bili tako dodani, so (Oracle, b.l., str. 5–6):

    - človeške aktivnosti (uporabniki lahko opravljajo naloge, ki jim jih dodeli poslovni proces);
    - začetek procesa (uporabnik je lahko iniciator procesa);
    - upravljanje s procesom (proces zahteva interakcijo z uporabnikom pred nadaljevanjem na naslednji aktivnosti);
    - tranzicija med avtomatiziranimi in človeškimi aktivnostmi (proces razume in upravlja izmenjavo človeških in avtomatizirani aktivnosti);
    - stopnjevalni mehanizmi (stopnjevalni mehanizmi se uporabljajo za stopnjevanje opravil, ki niso opravljena v predvidenem roku);
    - nominiranje (v nekaterih situacijah mora BPEL določiti izvajalca (nominiranca) v času izvajanja procesa);

- IBM in SAP sta, tako kot pri težavah s človeško interakcijo (BPEL4People), naletela na težave tudi pri upravljanju s kompleksnejšimi poslovnimi procesi, ki so sestavljeni in več modulov (podprocesov), in pri ponovni uporabi modulov (iz drugih procesov) v ostalih procesih. Predlagala sta dodatek, ki je, enako kot BPEL4People, v arhitekturi nameščen nivo višje od same BPEL-tehnologije. Podprocesi v osnovni različici BPEL-tehnologije morajo biti, če jih razvijalci želijo uporabljati še v katerem drugem procesu, definirani kot samostojni procesi in ne kot podproces drugega procesa, v sami logiki in dojemanju poslovnih procesov pa je to osnovna operacija, ki jo mora programski jezik, namenjen izključno izvajanju poslovnih procesov, omogočati. BPEL je v osnovni različici ne omogoča, zato IBM in SAP razvijeta dodatek, BPEL-SPE (Kloppmann, Koenig, Leymann, Pfau, Rickayzen, von Riegen, Schmidt & Trickovic, 2005, str. 4).

Pri razvoju tehnologije BPEL so vidna očitna pomanjkanja končne vizije o cilju, ki ga s tehnologijo želijo doseči. Dejstvo, da morajo ponudniki razvojnih ogrodij, ki za podporo poslovnim procesom uporabljajo tehnologijo BPEL, razvijati dodatke, ki omogočajo logično delovanje tehnologije in izpolnjevanje zahtev končnih uporabnikov.

4.1.2.2 Razvoj tehnologije Windows Workflow Foundation

Pri vseh dodatkih in poizkusih izboljšav pa ni nikoli sodelovalo podjetje Microsoft, ki je bilo pobudnik pri razvoju osnovne različice tehnologije BPEL. Razlog za to dobro pojasni Smith (Smith, 2005, str. 7–15). Po njegovem mnenju je do takšnega razvoja dogodkov na področju ponudnikov standardov in tehnologij za razvoj poslovnih programskih rešitev pripeljalo sledeče dejstvo: podjetje Microsoft je zgradilo svoj uspeh z zagotavljanjem operacijskih sistemov in pisarniških programskih orodij, dostopnih širši množici potrošnikov ter orodij za razvoj programskih rešitev za vsakdanjo uporabo in s tem doseglo prevladujoč položaj na tržišču, ki sega tudi v domene razvojnih tehnologij. Čeprav obstajajo podjetja, ki na trgu ogrodij in orodij za razvoj poslovnih programskih rešitev konkurirajo Microsoftu (SAP, Oracle, IBM itd.), morajo v sklopu lastnih proizvodov zaradi prevladujoče uporabe Microsoftovih operacijskih sistemov in orodij nuditi podporo tudi Microsoftovim razvojnim tehnologijam.

Microsoft je spoznal, da se smernice razvoja panoge razvojnih orodij za razvoj poslovnih programskih rešitev obračajo v smer, ko bodo orodja postala lažja za uporabo, razvoj programskih rešitev bo enostavnejši in ne bo potrebno poznavanje velikega števila programskih jezikov za razvoj programske rešitve, ki bo delovala na podlagi poslovnih procesov posamezne organizacije. Kot že omenjeno, je smernice v to smer obrnilo boljše zavedanje o pomembnosti poslovnih procesov in uvedba paradigme spletnih storitev. Potreba po učinkoviti orkestraciji spletnih storitev je tako kot druge tudi Microsoft spodbudila k razvoju lastne tehnologije za orkestracijo, XLANG, ki je, kot že opisano, tudi temelj BPEL-tehnologije.

Po vložkih v BPEL pa se je Microsoft od njega nekoliko oddaljil. Sami že imajo lasten operacijski sistem, ki je v množični uporabi (MS WINDOWS), orodja za pisarniško delo (MS Office), razvojno ogrodje (.NET) s skupkom pripadajočih standardov. Po uveljavitvi BPEL-

standarda je Microsoft spremenil svojo strategijo ter se odločil unovčiti sinergije, ki jih lahko doseže z integracijo podpore delovnim tokovom in poslovnim procesom v svoje obstoječe produkte, hkrati pa lahko tudi unovči znanje in izkušnje, ki izhajajo iz odzivov strokovne javnosti na uporabnost in pomanjkljivosti tehnologije BPEL. Odločili so se za razvoj Windows Workflow Foundation (WF), »izvajalnik delovnih tokov, programski model in nabor orodij za razvijalce, s katerimi bodo enostavno in hitro razvili programsko rešitev, ki bo temeljila na delovnem toku. Gre za nabor aktivnosti, ki koordinirajo delovanje uporabnikov in programske opreme« (Microsoft Redefines Workflow With Windows Workflow Foundation, 2005). Eric Rudder, tedanji podpredsednik oddelka za razvoj strežnikov in orodij pri Microsoftu, je ob predstavitvi WF poudaril, da bo ta podpiral delovne tokove, ki zahtevajo uporabniško (človeško) interakcijo ter tudi avtomatiziranje procesov, hkrati pa, da gre za preporod orodij za razvoj programske opreme, ki temelji na delovnih tokovih, saj bo ta tehnologija vgrajena v vse Microsoftove platforme, strežniške sisteme, ogrodja in orodja. »Ključno je, da v nasprotju z današnjimi standardnimi tehnologijami WF ne bo omejen in kompleksen, saj bo tehnologija že vsebovala svoj izvajalnik delovnih tokov in razvijalcem ne bo potrebno razvijati svojega, kot pri ostalih tehnologijah« (Microsoft Redefines Workflow With Windows Workflow Foundation, 2005). Microsoft v svojem prispevku ob predstavitvi poudari svojo strategijo, da bo WF postal del vseh njihovih proizvodov, vsako njihovo orodje bo podpiralo in spodbujalo uporabo WF-tehnologije.

Microsoft je predvidel, da so delovni tokovi in tehnologije, ki omogočajo gradnjo poslovnih programskih rešitev na njihovi osnovi, vedno bolj priljubljene. Z izkušnjami in odzivi na predhodne tehnologije so naredili naslednji korak z namenom, da bi omogočili razvoj kvalitetnejših programskih rešitev, razvitih v Microsoftovem ogrodju, z uporabo manjšega števila resursov pri razvoju. Cilj je bil, da se bodo lahko razvijalci osredotočili na poslovni proces in informatizacijo tega in ne na potrebo po implementaciji izvajalnika delovnih tokov in gradnji celotnega poslovnega procesa z uporabo programske kode. Razvidno je, da je Microsoft pri razvoju tehnologije uporabil vse negativne odzive na BPEL-tehnologijo, hkrati pa unovčil status vodilnega proizvajalca operacijskih sistemov in pisarniških orodij. Tehnologija Workflow Foundation je postala sestavni del vseh njihovih proizvodov, s takšnim pristopom pa so poizkušali na nek način tudi prisiliti razvijalce, da se poslužujejo njihove rešitve.

4.1.2.3 Ali Workflow Foundation podpira Business Process Execution Language?

David Chappell, ugledni IT-svetovalec, ki že več desetletji sodeluje s podjetjem Microsoft in je k znanosti prispeval že več strokovnih člankov in knjig, trdi, da je Microsoft obljubljal podporo BPEL-tehnologiji s strani lastne tehnologije Windows Workflow Foundation (WF) že vse od dne, ko je WF bil prvič predstavljen javnosti. Dejanska implementacija omenjene ideje pa se je zgodila šele v začetku leta 2007, kar je, tako trdi tudi Chappell (2007b), razmeroma pozno.

Same implementacije namreč ne gre razumeti kot odločitve Microsofta, da podpre razvoj poslovnih programskih rešitev z uporabo BPEL-tehnologije, temveč le kot medij, ki omogoča prenos programske kode v ogrodja, ki vsebujejo BPEL-izvajalnik, sam .NET pa to ni. BPEL-

koda oz. delovni tokovi so v okolju .NET prikazani v .NET podprti obliki (programski kodi) in ne v BPEL, enako velja ob uvozu BPEL-modelov, ti se namreč pretvorijo v WF-obliko. Pri podpiranju BPEL-a je Microsoft pri WF ubral enako strategijo, kot jo je pri orodju BizTalk Server, BPEL-tehnologijo dojema le kot medij, ki omogoča prenos delovnih tokov med posameznimi izvajalniki delovnih tokov in ne kot izvajalno tehnologijo.

Na podlagi kombinacije orodja BizTalk Server in BPEL-tehnologije je Chappel podal tudi mnenje, da ne gre pričakovati široke uporabe možnosti uvoza in izvoza BPEL-oblike, Microsoft naj bi se za ta korak odločil le za podpiranje tistih situacij, kjer je podpora oz. uporaba BPEL-tehnologije določena s politiko organizacije (Chappell, 2007b). Enakega mnenja je Michael Dortch, analitik podjetja RFG Group, ki trdi, da sama podpora BPEL-tehnologiji s strani Workflow Foundation ni noben velik korak v nobeno smer, saj so to naredili le z namenom, da omogočijo uvoz in izvoz poslovnih procesov, prenos med tehnologijama (McKendrick, 2007).

Chappellovo in Durtchovo mnenje dobita potrditev v prispevku podjetja Oracle (Oracle, 2006), ki obravnava tematiko pravilne izbire razvojnih standardov pri razvoju poslovnih programskih rešitev. Oracle poudarja, da je ključnega pomena pri zagotavljanju fleksibilnosti programskih rešitev in celotnih informacijskih sistemov usmerjenost v interoperabilnost, za doseganje tega pa je potrebno poslovne programske rešitve razvijati z uporabo standardov.

Oraclova orodja za razvoj poslovnih programskih rešitev temeljijo na javanskem ogrodju in sorodnih tehnologijah, razvijalci, ki uporabljajo Oraclova razvojna ogrodja bodo torej za razvoj novih rešitev uporabljali tehnologije, ki jih Oracle podpira. Z razvojem informacijske tehnologije in uvedbo paradigme SOA pa so se potrebe po poznavanju in povezovanju množice (tudi konkurenčnih) tehnologij med seboj povečale ter postale zahteva, ki jo mora izpolnjevati vsaka sodobna programska rešitev. Povezovanje različnih tehnologij med seboj pa ni omejeno le znotraj organizacije, temveč tudi in predvsem med posameznimi organizacijami, ki med seboj sodelujejo.

V prispevku se Oracle (2006) osredotoča predvsem na dejstvo, da zaradi potreb po vse večjem sodelovanju programskih rešitev med seboj ni več dovolj, da vsaka organizacija, ki proizvaja razvojna orodja, podpira le izbrane tehnologije, temveč je potrebno slediti standardom in komunicirati s konkurenčnimi organizacijami, ki podpirajo konkurenčne tehnologije. Ponudniki orodij morajo zagotavljati podporo konkurenčnim tehnologijam zaradi vrste razlogov, med katerimi so tudi sledeči primeri:

- uporaba BPEL-izvajalnika, implementiranega v ogrodju Java, za orkestracijo delovnega toka, ki za izvedbo uporablja storitve, razvite nekatere v ogrodju Java in nekatere v ogrodju .NET;
- razvoj nove .NET programske rešitve, ki bo pri izvajanju uporabljala tudi funkcionalnosti, ki so bile razvite v ogrodju Java in obratno;
- razvoj spletnih portalov, ki pri delovanju uporabljajo rešitve, razvite v .NET in Javanskem ogrodju;

- implementacija pametnega dokumenta (angl. *smart document solution*), kjer dokument zbirke Microsoft Office »kliče« BPEL-ov delovni tok ali spletno storitev, ki temelji na ogrodju Java ipd.

Pri omenjenih in podobnih primerih ne gre le za izmenjavo podatkov, temveč potrebo po zanesljivi in varni komunikaciji, ki podpira sinhrono in asinhrono sporočanje, pravilno preverjanje dovoljenj in omogočanje dostopov ter upravljanje s poslovnimi procesi in izmenjavo podatkov in sporočil (Oracle, 2006).

4.1.2.4 Trenutna situacija in smernice v prihodnosti

Microsoft v trenutnem obdobju načrtuje izdajo odprtokodne različice ogrodja .NET, ki se bo imenovala .NET Core. Vendar po pričevanjih uporabnikov razvijalskega foruma podjetja Github (Port Workflow Foundation to CoreFX/CoreCLR, 2016), trenutno ni sledi o uporabi WF v odprtokodni različici ogrodja .NET Core. Tako v dokumentaciji kot v samem jedru ogrodja CoreCLR ni zaznati sledi o WF. Pričevanja uporabnikov (razvijalcev) kažejo na to, da si želijo WF v odprtokodni verziji ogrodja, saj trdijo, da se trenutni trendi razvoja aplikacij in potreb uporabnikov gibljejo v smeri ponudbe tudi mobilnih poslovnih programskih rešitev, ki temeljijo na programski opremi na oblaku. Na istem spletnem mestu kot sogovornik deluje tudi oseba iz podjetja Microsoft, ki med uporabniki portala zbira izkušnje in kritike. V debati o tem ali naj se WF vgradi tudi v odprtokodno različico .NET Core je zaslediti trditve, ki kažejo na to, da je obravnavana implementacija v fazi razvoja ter da lahko uporabniki v prihodnosti to pričakujejo, ni pa posebej opredeljeno, kdaj bo do tega prišlo.

Po trditvah Microsofta je prvi korak oz. prioriteta razvoja v smeri prenosa WF na ostale .NET sorodne produkte, kot je .NET Core, implementacija WF v Microsoft Nano Server. Nano Server je operacijski sistem, ki je bil razvit z namenom izvajanja programskih rešitev, ki jih Microsoft poimenuje "born-in-the-cloud", kar v prevodu pomeni razvitih za delovanje v oblaku. Poslanstvo operacijskega sistema Nano Server, ki izhaja iz in temelji na operacijskem sistemu Windows Server, je slediti trendom pri razvoju in uporabi programskih rešitev, kjer se za izvajanje programskih rešitev v oblaku zahteva hitro odzivnost, agilnost ter uporabo čim manjšega števila resursov (kot npr. pasovna širina, procesna moč ipd.).

Dejstvo pa je, da se tudi na Microsoftovem razvojnem portalu MSDN (Workflow in the Future, 2016) pojavljajo vprašanja, kakšna je prihodnost tehnologije Workflow Foundation. Uporabniki, ki sodelujejo v temi namreč izpostavljajo dejstvo, da je kakršno koli spreminjanje in opravljanje izboljšav na tehnologiji na neki mrtvi točki že več let (od leta 2013), prav tako je tudi zadnja objava na krovni spletni strani tehnologije bila objavljena leta 2013. Vlada sicer splošno prepričanje, da je tehnologija še vedno aktualna, vendar pa se podjetje Microsoft vzdržuje vsakršnega komentarja na temo razvoja ali opustitve tehnologije Workflow Foundation. Vendar, če gre verjeti nekaterim uporabnikom, naj bi Microsoft tudi v prihodnosti podpiral tehnologijo, saj je še vedno sestavni del večine njihovih proizvodov.

Iz zgornjih vrstic je mogoče tudi sklepati, da bo razvoj prihodnih jedrnih tehnologij šel odločneje v smer podpore poslovnih programskih rešitev za uporabo na mobilnih napravah, hkrati pa bo še izrazitejši poudarek na vložkih v jedrne tehnologije, ki bodo omogočale razvoj poslovnih programskih rešitev v okviru storitev v oblaku od samega razvoja do implementacije in tudi uporabe, v kombinaciji z učinkovito podporo za mobilne naprave.

Poudariti je potrebno tudi, da je razvoj BPEL-tehnologije v rokah neprofitne organizacije OASIS, zadnja nadgradnja tehnologije pa se je zgodila v mesecu aprilu leta 2007 (OASIS Web Services Business Process Execution Language (WSBPEL) TC, 2016**).** Sicer gre v primeru BPEL-tehnologije za popolnoma drugačno situacijo kot v primeru tehnologij organizacije Microsoft. V primeru BPEL gre namreč za standard, ki ga mora, kot že poudarjeno, vsak izmed ponudnikov orodij in tehnologij za razvoj poslovnih programskih rešitev dokončno opremiti s svojim izvajalnikom in implementirati v razvojna orodja in ogrodje. Za razvoj tehnologij torej vsak izmed ponudnikov (IBM, Oracle, SAP itd.) skrbi sam. Oracle v svoji knjigi (Saraswathi & Singh, 2013, str. 268) BPEL še vedno opisuje kot standard, ki je ključen za izvedbo delovnih tokov. Vendar pa, tako v knjigi kot tudi na nekaterih forumih, med katerimi je tudi IBM-ov DeveloperWorks, v temi o IBM WebSphere (BPMN vs BPEL, 2016), ter enako na portalu Oracle Technology Network (BPM vs BPEL Oracle 11g, 2016), na temo prihodnosti obstajajo nekatera razmišljanja v zvezi s tehnologijo BPMN 2.0, ki bi lahko v prihodnosti izrinila BPEL. Dejstvo je, da je BPMN 2.0 novejša tehnologija, ki omogoča tudi izvajanje delovnih tokov, ki so v sklopu te tehnologije tudi predstavljeni v obliki notacije in ne samo v obliki programske kode. Sicer je to mogoče tudi z uporabo BPEL-tehnologije, vendar v kombinaciji z implementacijami posameznih ponudnikov razvojnih orodij. Uporabniki so v sklopu teme razpravljali o tem, ali lahko BPMN različice 2.0 nadomesti BPEL, skupni zaključek pa je bil, da je sicer res tehnologija BPMN 2.0 boljša za prikaz poslovnih procesov, omogoča tudi izvedbo delovnih tokov, vendar vodilni ponudniki razvojnih orodij za izvajanje delovnih tokov še vedno uporabljajo, ter bodo tudi v prihodnosti, tehnologijo BPEL, BPMN pa uporabljajo v večini primerov le za modeliranje poslovnih procesov. BPEL je namreč veliko boljša tehnologija ob kritičnih dogodkih v delovnem toku, hkrati pa je tesno povezana s spletnimi storitvami, ki so trenutno še vedno prevladujoča arhitektura oz. paradigma razvoja poslovnih programskih rešitev (BPMN 2.0 vs BPEL – Oracle BPM vs SOA Suite, 2016**).**

4.2 Primerjava Windows Workflow Foundation z Business Process Execution Language

Evalvacija tehnologij, ki omogočajo gradnjo sodobnih poslovnih programskih rešitev, je zapletena, ker je odločitev o tem, katero tehnologijo uporabiti, pogosto pogojena tudi z znanjem razvijalcev, odvisna je od obstoječih programskih rešitev in kompatibilnosti le-teh z načrtovano novo rešitvijo, ceno itd. Kljub temu pa obstaja metodologija, ki omogoča evalvacijo tehnologij, ki omogočajo izvajanje delovnih tokov, to so vzorci delovnih tokov (angl. *Workflow Patterns*).

Pojem oz. idejo o gradbenih vzorcih je prvič predstavil leta 1978 Alexander v svojem delu »A Pattern Language«, kjer obravnava vzorce fizičnih struktur, ki se uporabljajo v gradbenih in arhitekturnih delih. Trdi, da je mogoče urbanistične težave reševati na podlagi usklajenih vzorcev, ki omogočajo optimalno uporabo gradbenih posegov v okolju, v knjigi uvaja t. i. vzorčni jezik (angl. *Pattern Language*). Vsak vzorec opisuje težavo, ki se pojavlja v vsakdanjem življenju ob načrtovanju in gradnji objektov ter nato predlaga optimalno rešitev, najboljšo prakso (Alexander, 1978, str. »xi«). Čez skoraj dve desetletji so leta 1995 Gamma, Helm, Johnson in Vissides (1995) objavili knjigo Design Patterns: Elements of Reusable Object-Oriented Software, kjer so uporabo Alexandrovih konceptov transponirali na uporabo v logičnih arhitekturah, v načrtovanje programskih rešitev. Uporaba vzorcev pri načrtovanju programskih rešitev je od tistega trenutka dalje začela pridobivati vedno več pozornosti v informacijski tehnologiji, vključno pri gradnji poslovnih procesov in storitveno usmerjenih arhitekturah. Vzorci, kot jih definirajo Gamma et al. opisujejo abstraktne in elegantne rešitve za pogoste težave, ki se pojavljajo pri gradnji oz. razvoju programskih rešitev. Vzorci so z uporabo hevristike pridobljene najboljše prakse. Z uporabo hevristike in izhajanja iz vzorcev uporabe je mogoče doseči krajši čas razvoja programskih rešitev, hkrati pa tudi kvaliteto in trdnost izdelkov (Gamma et al., 1995, str. 1–2). Vendar pa so Gamma et al. v svojem katalogu vzorcev sistematično definirali le 23 načrtovalskih vzorcev (ang. design patterns), ki opisujejo manjše ponavljajoče se interakcije v objektno usmerjenih sistemih. Načrtovalski vzorci naj bi v osnovi omogočali neodvisnost od implementacijske tehnologije ter istočasno neodvisnost od osnovnih zahtev domene, v kateri bodo implementirani (Van der Aalst, ter Hofstede, Kiepuszewski & Barros, 2003, str. 5–51). Leta 2003 so van der Aalst et al. idejo Gamma et al. (1995, str. 1–2) nadgradili z željo, da bi vzorce uporabili za druge, primerjalne potrebe. Namesto za prikaz najboljših praks in reševanj pogostih težav pri razvoju, bi vzorce uporabili za primerjavo jezikov za razvoj programskih rešitev, ki temeljijo na poslovnih procesih. Vzorci bi služili kot merilo za primerjavo možnosti izražanja posameznih programskih jezikov (Van der Aalst et al., 2003, str. 5–51).

Vzorci delovnega toka imajo tudi to prednost, da izražajo strokovno primerjavo med tehnologijami, v nasprotju s primerjavami, opravljenimi s strani neodvisnih svetovalnih družb. Te namreč pri primerjavi različnih tehnologij in orodij upoštevajo v večji meri tehnične in tržne vidike, samih tehnologij, na katerih obravnavana orodja, ki so prisotna na trgu temeljijo pa ne podrobneje obravnavajo (Van der Aalst et al., 2003, str. 5–51).

4.2.1 Opis metodologije preizkusa

Metodologija evalvacije tehnologij za izvajanje delovnih tokov zajema preizkušanje zmogljivosti in konstruktov, ki bi morali biti prisotni v programskih jezikih za modeliranje in izvajanje delovnih tokov.

Van der Aalst, Barros, ter Hofstede in Kiepuszewski (2000, str. 18–29) so v raziskavi z naslovom »Advanced Workflow Patterns« identificirali prve vzorce delovnega toka, natančneje vzorce za nadzor toka podatkov (angl. *control-flow*), kasneje so isti avtorji v delu »Workflow

Patterns« (2003, str. 5–51) predstavili prvi katalog dvajsetih vzorcev za nadzor toka podatkov. Objava tega prispevka je sprožila zanimanje med raziskovalci, ki so svoje raziskovanje usmerili v dve smeri; nekateri so želeli vzorce uporabiti za oblikovanje okvirov za razumevanje zahtev pri nadzoru toka podatkov, drugi pa so hoteli vzorce uporabiti za vrednotenje zmogljivosti raznih programskih jezikov, ki služijo modeliranju in izvajanju delovnih tokov, to pa je področje, ki nas pri izdelavi te naloge zanima.

Z nadaljnjimi raziskavami na področju vrednotenja zmogljivosti z vzorci (Russel, ter Hofstede, Edmond & van der Aalst, 2005, str. 353–368) so raziskovalci poudarili, da je smiselno pripraviti kataloge vzorcev tudi za dodatne vidike delovanja tehnologij, kot npr. podatkovni vidik (angl. *data perspective*), vidik resursov (angl. *resource perspective*) ter vzorci za prestrezanje izjem (angl. *exception handling*). Poleg vzorcev za nadzor toka podatkov pa so ostali katalogi vzorcev (podatkovni vidik, vidik resursov, prestrezanje izjem) odvisni tudi od preostalih delov arhitekture programskih rešitev (ogrodje in spremljajoče tehnologije). Vzorci za nadzor toka podatkov pa so tisti, ki podajo oceno o zmogljivosti posamezne tehnologije za modeliranje in izvajanje delovnih tokov (Lenhard, 2011, str. 28).

Katalog vzorcev za nadzor toka podatkov danes zajema 43 vzorcev; poleg prvotnih 20, so Russel, ter Hofstede, van der Aalst in Mulyar (2006) dodali še 23 vzorcev, vsi so predstavljeni v Prilogi 2.

Našteti vzorci pa se sicer delijo tudi v različne skupine, te so:

1. osnovni vzorci: WCP-1–WCP-5;
2. napredni vzorci z razcepi in sinhronizacijami: WCP-6–WCP-9; WCP-28–WCP-33; WCP-37, WCP-38; WCP-41, WCP-42;
3. vzorci s podporo več instancam aktivnosti: WCP-12–WCP-15; WCP-34–WCP-36;
4. vzorci za upravljanje s stanji: WCP-16–WCP-18; WCP-39, WCP-40;
5. preklicni vzorci: WCP-19 in WCP-20; WCP-25–WCP-27;
6. ponovitveni vzorci: WCP-10; WCP-21 in WCP-22;
7. zaustavitveni vzorci: WCP-11; WCP-43;
8. sprožilni vzorci: WCP-23 in WCP-24.

Pri ugotavljanju podpore, ocenjevanju in analiziranju tehnologij za modeliranje in izvajanje delovnih tokov so se vzorci delovnih tokov uveljavili tako v akademski kot tudi v poslovni sferi kot orodje za sprejemanje odločitev in analize (Zapletal et al., 2009, str. 1), po izvedbi raziskave pa smo zasledili le dve študiji, ki ocenjujeta tehnologijo Workflow Foundation. Študij, ki obravnavajo tehnologijo BPEL, je sicer več, takšni, ki zajemata celoten katalog vzorcev za nadzor toka podatkov ter sta primerljivi s študijama o Workflow Foundation pa sta ravno tako dve. Obravnavane študije so sicer opravljene po enaki metodologiji, torej na podlagi vzorcev delovnih tokov:

Workflow Foundation:

- an Analysis of Windows Workflow's Control-Flow Expressiveness (Zapletal et al., 2009, str. 200–209) in
- A Pattern-based Analysis of WS-BPEL and Windows Workflow (Lenhard, 2011);

Business Process Execution Language:

- Workflow Control-Flow Patterns (Russel et al., 2006);
- A Pattern-based Analysis of WS-BPEL and Windows Workflow (Lenhard, 2011).

4.2.2 Analiza študij in primerjava rezultatov

Kot je razvidno iz Tabele 3, se študije med seboj razlikujejo tudi po obdobjih, kdaj so bile narejene. Pomembno je poudariti, da sta se pri obeh tehnologijah študije opravljale na dveh različicah:

Tabela 3: Različice tehnologij v obravnavanih študijah

ŠTUDIJA	BPEL	Workflow Foundation
Russel et al. (BPEL, 2006); Zapletal et al. (WF, 2009)	1.1	3.5
Lenhard (BPEL, WF, 2011)	2.0	4.0

4.2.2.1 Business Process Execution Language

V Tabeli 4 so predstavljeni rezultati obeh raziskav, ki obravnavata BPEL-tehnologijo, v tabeli so vključeni tudi primeri izvajanja vzorcev na BPEL-izvajalnikih različnih ponudnikov orodij za razvoj poslovnih programskih rešitev; Oracle in IBM na različici BPEL 1.1 in Sun na različici BPEL 2.0.

Za razumevanje rezultatov je potrebno razumeti način ocenjevanja, ki pa se med študijama nekoliko razlikuje, vendar kot poudarja Lenhard (2011, str. 39), še vedno ohranja primerljivost raziskav. Dejstvo pa je, da je natančnost podajanja ocen subjektivna, saj je odvisna tudi od strokovnosti načrtovalca programske rešitve (Van der Aalst et al., 2003, str. 45).

Obe študiji rezultate vrednotita z enakimi simboli (+; +/–; –), pri čemer pri Russel et al. (2006) pomeni (Van der Aalst et al., 2003, str. 45):

- +: vzorec je podprt in enostavno razvit z uporabo grafičnega vmesnika orodja oz. programskega ukaza s spoštovanjem vodil posameznega vzorca, ki temeljijo na notaciji Petri Net;

- +/–: vzorec je delno podprt, kar pomeni, da je za implementacijo vzorca potrebna kombinacija vzorcev oz. programskih ukazov, vendar je ta še vedno mogoča in ne zahteva dodatnega prilagajanja s programsko kodo. Implementacija z upoštevanjem osnovnih Petri Net vodil ni mogoča;
- –: vzorec ni podprt. To ne pomeni, da implementacija ni mogoča, vendar zahteva zahtevno uporabo t. i. špageti programske kode oz. izdelavo zapletenih procesnih modelov.

Hkrati pa študija za vrednotenje vsakemu posameznemu vzorcu definira potrebne kriterije, ki opredeljujejo podprtje vsakega vzorca posebej, v nasprotju s študijo Lenharda (2011, str. 41), kjer kriterije za podprtje vzorcev definira krovno, za vse vzorce enake:

- +: vzorec je podprt, če izpolnjuje sledeče kriterije:
  - ko so izpolnjeni vsi kontekstualni kriteriji vzorca;
  - doseženi vsi vidiki načrtovanja procesa;
  - vzorec mora biti samostojen konstrukt;
- +/–: vzorec je delno podprt, ko:
  - največ eden izmed kontekstualnih kriterijev ni dosežen;
  - največ eno izmed vodil načrtovanja procesa ni doseženo;
  - je sestavljen iz največ dveh konstruktov;
- –: vzorec ni podprt, ko:
  - implementacija ne izpolnjuje zahtev vzorca;
  - je sestavljen iz več kot dveh konstruktov.

Russel et al. (2006) v študiji preverjajo osnovno različico BPEL 1.1, hkrati pa so v raziskavo vključili tudi Oracle BPEL in WebSphere BPEL, ki sta implementaciji Oracla in IBM-a. Kot je bilo že v prejšnjem poglavju omenjeno, je pri BPEL-tehnologiji posebnost ta, da gre za standard, vendar pa mora vsak ponudnik orodij za razvoj poslovnih programskih rešitev priskrbeti svoj izvajalnik za izvajanje in uporabo BPEL-tehnologije. V raziskavi je razvidno (Tabela 4), da implementacija lastnega izvajalnika predstavlja tudi razlike pri zmogljivostih tehnologije v kombinaciji z različnimi BPEL-izvajalniki. Takšen primer predstavljata vzorca WCP-13 in WCP-14, ki predstavljata vzorec, ko je število ponovitvenih instanc aktivnosti znano ob načrtovanju oz. zagonu delovnega toka. Oraclova implementacija BPEL-tehnologije omogoča izvajanje omenjenih vzorcev, medtem ko IBM-ova in osnovna tega ne podpirata. V nasprotju pa Oracle v svojem BPEL-u ne podpira vzorca št. 17, prepletenega vzporednega usmerjanja. Oracle je tehnologijo BPEL v obravnavanem obdobju oplemenitil z nekaterimi zmogljivostmi, ki so nato postale standardne s prihodom različice BPEL 2.0.

V primerjavi med raziskavama je razvidno, da so rezultati med njima razmeroma podobni, razvidne pa so tudi razlike, ki so nastale zaradi opravljanja raziskave na novejši različici tehnologije BPEL, 2.0. V kategoriji »Napredni vzorci z razcepi in sinhronizacijami« je opaziti, da tehnologija BPEL 2.0 podpira ali delno podpira le 3 vzorce od skupnih 14, različica 1.1 pa 5 od skupnih 14. Glavni razlog, da starejša različica tehnologije podpira oz. delno podpira večje

število vzorcev od novejše, tiči v kriterijih, ki so bili uporabljeni za vrednotenje podpore vzorcem. Russel et al. (2003, str. 75) pri vzorcih WCP-41 in WCP-42 dovoljujejo uporabo programerskih veščin (poseg s programsko kodo) za prilagoditev vzorca, medtem ko Lenhard (2011, str. 59) prilagajanja s programerskimi posegi ne dovoljuje. Podatek, da je večina vzorcev nepodprtih s strani BPEL v obravnavani kategoriji, temelji na dejstvu oz. pomanjkljivosti BPEL-tehnologije. Vsi vzorci, ki niso podprti (WCP-8, WCP-9; WCP-28–WCP-33 in WCP-38) namreč temeljijo na združevanju več vej procesa v eno, brez sinhronizacije. To pomeni, da se v primeru izvajanja aktivnosti po spojitvi v eno na več vzporednih vejah hkrati v sledeči aktivnosti po združitvi izvedejo za vsako vejo posebej, naenkrat se na eni veji izvaja več niti delovnega toka. To pa je omejitev BPEL-tehnologije – ne omogoča namreč izvajanja več niti delovnega toka hkrati na isti veji procesa (Ter Hofstede et al., 2010, str. 397).

Kategorija »Vzorci s podporo več instancam« v primerjavi s predhodno obravnavano kategorijo prikazuje zrcalno sliko, razviden je namreč napredek tehnologije v različici 2.0. V primerjavi s predhodno verzijo (1.1), kjer je podprt ali delno podprt le eden (1) od sedmih (7) vzorcev, se v Lenhardovi (2011) raziskavi to spremeni, podprtih oz. delno podprtih je pet (5) od sedmih (7) vzorcev. BPEL 2.0 uvaja ukaz »forEach«, ki omogoča izvajanje zanke za število ponovitev, določeno ob zagonu delovnega toka. Pred prihodom tehnologije BPEL 2.0 je funkcijo omogočal le Oracle, ki je z razvojem lastnega dodatka – ukaza »flowN« – želel odpraviti pomanjkljivost. Enak razlog je ključen tudi v kategoriji »preklicni vzorci«, kjer so z uporabo BPEL 2.0 podprti ali delno podprti vsi vzorci, v primerjavi s predhodno raziskavo, kjer vzorca WCP-26 in WCP-27 nista bila podprta, z izjemo Oraclove različice zaradi odsotnosti ukaza »forEach«.

Napredek BPEL 2.0 v primerjavi s predhodno različico je viden tudi pri naslednji kategoriji vzorcev, »Vzorci za upravljanje s stanji«, kjer so bili ob prvi raziskavi podprti ali delno podprti štirje od petih vzorcev, pri BPEL 2.0 pa so podprti oz. delno podprti vsi vzorci, razlog je v implementaciji funkcije »OnAlarm« v novejši različici. Vzorec WCP-18 pri Russelovi (2003) raziskavi ni omogočal implementacije z uporabo BPEL-tehnologije, medtem ko funkcija »OnAlarm« to delno omogoča. Razlike pri vzorcih WCP-39 in WCP-40 so nastale zaradi razlik v kriterijih, vzorca namreč zahtevata izgradnjo dveh konstruktov za implementacijo vzorca, Lenhard (2011) to kategorizira kot delno podporo, Russel (2003) pa zaradi individualnih kriterijev za posamezne vzorce to označuje kot popolno podporo.

Kategorija »ponovitveni vzorci« prikazuje popolnoma enako stanje med obema raziskavama, podprt je samo vzorec WCP-21, medtem ko vzorca WCP-10 in WCP-22 nista podprta zaradi narave tehnologije BPEL. Tehnologija namreč temelji na blokovni strukturi, to pa onemogoča gradnjo in opis zank z več kot enim vhodom in izhodom. Sama logika tehnologije ne dovoljuje uporabe arbitrarnih ciklov in rekurzije (Ter Hofstede et al., 2010, str. 394).

Novost različice BPEL 2.0 je tudi upravitelj zaustavitev delovnega toka, »FaultHandler«, ki omogoča podprtje vseh »zaustavitvenih vzorcev«, v primerjavi s prvo raziskavo, kjer je bil podprt le vzorec št. 11, medtem ko pa je stanje v kategoriji »sprožilni vzorci« ostalo med

raziskavama enako, enostavna možnost za podprtje vzorca WCP-23, žal, ne obstaja (Lenhard, 2011, str. 75).

Tabela 4: Rezultati BPEL – Russel et al. in Lenhard

	Russel et al. (2006)			Lenhard (2011)	
VZOREC	**BPEL**	**Oracle BPEL**	**WebSphere BPEL**	**BPEL**	**SunBPEL**
Osnovni vzorci					
WCP-1	+	+	+	+	+
WCP-2	+	+	+	+	+
WCP-3	+	+	+	+	+
WCP-4	+	+	+	+	+
WCP-5	+	+	+	+	+
Napredni vzorci z razcepi in sinhronizacijami					
WCP-6	+	+	+	+	+/–
WCP-7	+	+	+	+	+/–
WCP-8	–	–	–	–	–
WCP-9	–	–	–	–	–
WCP-28	–	–	–	–	–
WCP-29	–	–	–	–	–
WCP-30	–	–	–	–	–
WCP-31	–	–	–	–	–
WCP-32	–	–	–	–	–
WCP-33	–	–	–	–	–
WCP-37	+	+	+	+	–
WCP-38	–	–	–	–	–
WCP-41	+/–	+/–	+/–	–	–
WCP-42	+/–	+/–	+/–	–	–
Vzorci s podporo več instancam aktivnosti					
WCP-12	+	+	+	+	+/–
WCP-13	–	+	–	+	+/–
WCP-14	–	+	–	+	–
WCP-15	–	–	–	–	–
WCP-34	–	–	–	+/–	–
WCP-35	–	–	–	+	–
WCP-36	–	–	–	–	–
Vzorci za upravljanje s stanji					
WCP-16	+	+	+	+	+
WCP-17	+/–	–	+/–	+/–	–
WCP-18	–	–	–	+/–	+/–
WCP-39	+	+	+	+/–	–
WCP-40	+	–	+	+/–	–
Preklicni vzorci					

se nadaljuje

nadaljevanje

	Russel et al. (2006)			Lenhard (2011)	
VZOREC	**BPEL**	**Oracle BPEL**	**WebSphere BPEL**	**BPEL**	**SunBPEL**
Preklicni vzorci					
WCP-19	+	+	+	+/–	+/–
WCP-20	+	+	+	+	+
WCP-25	+/–	+/–	+/–	+/–	+/–
WCP-26	–	+	–	+/–	–
WCP-27	–	–	–	+	–
Ponovitveni vzorci					
WCP-10	–	–	–	–	–
WCP-21	+	+	+	+	+
WCP-22	–	–	–	–	–
Zaustavitveni vzorci					
WCP-11	+	+	+	+	+
WCP-43	–	–	–	+	+
Sprožilni vzorci					
WCP-23	–	–	–	–	–
WCP-24	+	+	+	+	+

Vir: Prirejeno po N. Russel et al., Workflow Control-Flow Patterns: A Revised View, 2006, str. 79; J. Lenhard, A Pattern-based Analysis of WS-BPEL and Windows Workflow, 2011, str. 77.

4.2.2.2 Windows Workflow Foundation

Poudariti moramo, da so kriteriji ocenjevanja popolnoma enaki kot pri študijah BPEL-tehnologije. Med seboj se razlikujejo na enak način, Zapletal et al. (2009) v svoji študiji upoštevajo način, ki je uporabljen v Russel et al. (2006), vključno z zahtevami za izpolnjevanje kriterijev, ki se razlikujejo od vzorca do vzorca, ravno tako je Lenhard (2011) v svoji raziskavi uporabil enake kriterije kot pri raziskavi BPEL-tehnologije.

Omeniti moramo, da se raziskavi razlikujeta tudi v različici tehnologije, ki je bila aktualna v času raziskave, Zapletal et al. (2009) opravljajo raziskavo z uporabo različice tehnologije Workflow Foundation 3.5, medtem ko Lenhard za svojo raziskavo uporabi Workflow Foundation 4.0. Ključna razlika med obema je v tem, da je Workflow Foundation 3.5 omogočal dva načina izvajanja delovnih tokov in razvijanja programskih rešitev, zaporedni način in način avtomata stanj, medtem ko različica 4.0 avtomata stanj ne podpira več, temveč le zaporedni način ter hkrati razvoj v obliki diagrama toka podatkov. Zapletal et al. (2009) v svoji raziskavi predstavljajo rezultate, opravljene z uporabo zaporednega načina, hkrati pa tam, kjer sam zaporedni način ni dovolj oz. vzorca ne podpira, uporabljajo avtomat stanj (A. S. v Tabeli 5), če je to mogoče (če ta izvajanje vzorca podpira).

Po pregledu Tabele 5 je razvidno, da se pri osnovnih vzorcih obe raziskavi skladata med seboj, v sklopu naprednih vzorcev pa so vidna prva odstopanja med njima. Pri vzorcih WCP-6 in WCP-7 sta odstopanji rezultat razlik nadgradnje tehnologije. V primerjavi z različico 3.5, ki je bila aktualna v času prve raziskave, različica 4.0 ne vsebuje več nekaterih aktivnosti, med njimi tudi tiste (ConditionActivityGroup), ki je omogočala implementacijo vzorcev WCP-6 in WCP-7. Podpora je zato delna, saj je še vedno mogoča z uporabo kombinacij aktivnosti, hkrati pa Lenhard (2011) ocenjuje, da podpora ni celotna zaradi uporabe dveh konstruktov za podporo vzorcu. Večje odstopanje pa je prisotno pri vzorcu WCP-37, kjer se trditvi avtorjev med seboj zelo razlikujeta. Lenhard (2011) trdi, da opisovanje nestrukturiranih procesnih modelov, kjer se združi več aktivnih vej delovnega toka, z uporabo tehnologije Workflow Foundation ni mogoče oz. ni podprto. Združevanje več vej in podvej nestrukturiranega procesa ob hkratnem upoštevanju pogojev združevanja na način, kot jih zahteva vzorec WCP-37 (spremenljivk tipa DA/NE – Boolean), ni mogoče. Hkrati pa Zapletal et al. (2009) trdijo, da je to delno mogoče, vendar je potrebno programersko poseganje v obstoječe osnovne aktivnosti tehnologije, »samo ob prisotnosti izključno strukturiranih procesnih modelov«. Z upoštevanjem obeh trditev lahko zaključimo, da vzorec v obliki, kot je zahtevana s strani Russla (2006, str. 69) ni podprt. Obe raziskavi sicer podpirata šest (WCP-37 ni upoštevan) od skupno štirinajstih vzorcev v tej skupini.

Skupina »Vzorci s podporo več instancam aktivnosti« ne predstavlja razlik med raziskavama, saj so po rezultatih obeh podprti vsi vzorci, medtem ko je nekaj posebnosti v skupini vzorcev za upravljanje stanj. Prva raziskava (2009) z uporabo sekvenčnega načina podpira tri od skupno petih vzorcev te skupine, medtem ko je z uporabo avtomata stanj mogoča podpora celotni skupini vzorcev, enako velja za drugo raziskavo (2011). Obstajajo pa razlike v vrednotenju rezultatov, ki so rezultat drugačnih kriterijev ocenjevanja ter tudi razlik pri implementaciji vzorca. Pri primeru WCP-16 so avtorji obravnavanih raziskav izbrali različna načina za implementacijo, vendar pa bi tudi ob izbiri enakega prišlo do odstopanja pri vrednotenju. Lenhard namreč za uporabo dveh konstruktov, kot že omenjeno, ocenjuje podprtost vzorca kot delno, Zapletal et al. (2009, str. 200–209) pa so za implementacijo uporabili dva konstrukta, kot Lenhard (2011). Vzorca WCP-17 in 18 prikazujeta, da je Microsoft kljub prenehanju uporabe avtomata stanj tehnologiji Workflow Foundation omogočil drug način za implementacijo vzorcev, kjer je stanje potrebno. Pri prvi raziskavi je implementacija bila mogoča le ob uporabi avtomata stanj, pri drugi pa je bilo to mogoče izvesti tudi brez te, z uporabo novejših pristopov.
Ocene podpore vzorcev WCP-39 in WCP-40, tako kot ocene vzorcev WCP-19, WCP-25 in WCP-26 v skupini »Preklicni vzorci«, se med seboj razlikujejo izključno zaradi razlik v kriterijih ocenjevanja, medtem ko je odstopanje med raziskavama pri vzorcu WCP-27 rezultat nadgradnje tehnologije. Prisilno dokončanje izvajanja aktivnosti procesa pri različici 3.5 ni bilo mogoče, Workflow Foundation 4.0 pa to možnost ima.

Skupina »Ponovitveni vzorci« pri vzorcu WCP-10 nakazuje na možnost, ki je omogočala implementacijo z uporabo avtomata stanj na enostaven način, ta pri različici 4.0 zahteva kombinacijo več aktivnosti in je za implementacijo zelo zapletena, vendar mogoča. Rezultati ostalih vzorcev skupine so med raziskavama enaki.

»Zaustavitveni vzorci« ne predstavljajo razlik med raziskavama, omeniti pa je potrebno razhajanje pri vzorcu WCP-23 v skupini »Sprožilni vzorci«. Novejša izvedba tehnologije namreč vzorca ne podpira več, je pa prisotna tudi napaka v raziskavi Zapletal et al. (2009, str. 200–209), kjer raziskovalci trdijo, da je implementacija mogoča tudi z uporabo zaporednega načina. Microsoft v svoji dokumentaciji namreč način izvajanja takšnega vzorca omejuje izključno na uporabo avtomata stanj (Windows Workflow Tutorial, 2016), Workflow Foundation 4.0 tega ne podpira.

Tabela 5: Rezultati BPEL – Zapletal et al. in Lenhard

	Zapletal et al. (2009)		Lenhard (2011)
VZOREC	**WF**	**WF (A. S.)**	**WF**
Osnovni vzorci			
WCP-1	+	+	+
WCP-2	+	+	+
WCP-3	+	+	+
WCP-4	+	+	+
WCP-5	+	+	+
Napredni vzorci z razcepi in sinhronizacijami			
WCP-6	+	+	+/–
WCP-7	+	+	+/–
WCP-8	–	–	–
WCP-9	+/–	+/–	+/–
WCP-28	–	–	–
WCP-29	+	+	+
WCP-30	+/–	+/–	+/–
WCP-31	–	–	–
WCP-32	+	+	+
WCP-33	–	–	–
WCP-37	+/–	+/–	–
WCP-38	–	–	–
WCP-41	–	–	–
WCP-42	–	–	–
Vzorci s podporo več instancam aktivnosti			
WCP-12	+	+	+
WCP-13	+	+	+
WCP-14	+	+	+
WCP-15	–	–	–
WCP-34	+/–	+/–	+/–
WCP-35	+	+	+
WCP-36	–	–	–
Vzorci za upravljanje s stanji			

se nadaljuje

nadaljevanje

	Zapletal et al. (2009)		Lenhard (2011)
VZOREC	**WF**	**WF (A. S.)**	**WF**
Vzorci za upravljanje s stanji			
WCP-16	+	+	+/–
WCP-17	–	+	+
WCP-18	–	+	+/–
WCP-39	+	+	+/–
WCP-40	+	+	+/–
Preklicni vzorci			
WCP-19	+	+	+/–
WCP-20	+	+	+
WCP-25	+	+	+/–
WCP-26	+	+	+/–
WCP-27	–	–	+
Ponovitveni vzorci			
WCP-10	–	+	–
WCP-21	+	+	+
WCP-22	–	–	–
Zaustavitveni vzorci			
WCP-11	+	–	+
WCP-43	–	+	+/–
Sprožilni vzorci			
WCP-23	+	+	–
WCP-24	+	+	+

Vir: Prirejeno po M. Zapletal et al., An Analysis of Windows Workflow's Control-Flow Expressiveness, 2009, str. 200–209; J. Lenhard, A Pattern-based Analysis of WS-BPEL and Windows Workflow, 2011, str. 77.

4.2.3 Primerjava tehnologij na podlagi rezultatov raziskav

Russel et al. (2006) ugotavljajo, da je BPEL primerna tehnologija, ko je potrebno obvladovanje strukturiranih sinhronizacij, zaradi svoje blokovne strukture, obstaja pa omejitev, ko je potrebna večnitna obravnava iste veje procesa. BPEL 1.1., ki je bil aktualna različica tehnologije v času raziskave, namreč ni omogočal večnitnega istočasnega izvajanja posamezne veje poslovnega procesa, to je glavni razlog, da večina vzorcev z razcepi in sinhronizacijami ne omogoča enostavne implementacije. Kot smo že omenili, avtorji raziskave ugotavljajo, da je Oracle svojo različico BPEL-izvajalnika oplemenitil z dodatnimi zmožnostmi, ki omogočajo izvajanje zank s poljubnim številom ponovitev. Dodatna omejitev BPEL-tehnologije, ki se izrazito prikazuje iz raziskave, je obvladovanje več istočasnih instanc delovnega toka in spojitev le-teh, hkrati pa je BPEL-tehnologija ena redkih, ki omogoča delovanje na podlagi t. i. žetonov (angl. *token-based*) oz. signalov, ki so potrebni za nadaljevanje izvajanja delovnega toka. Iz raziskave pa je jasno

razvidna strategija razvoja BPEL-tehnologije, saj omogoča ponudnikom razvojnih orodij gradnjo BPEL-izvajalnikov po meri, to pa omogoča delno izolacijo pomanjkljivosti in nadgradnjo.

Od vseh **43 vzorcev** so Russel et al. (2006) ugotovili, da BPEL 1.1 podpira ali delno podpira **21 vzorcev**, kar predstavlja **48,84 odstotka vseh vzorcev** (Tabela 6).

Tabela 6: Podpora skupinam vzorcev BPEL 1.1

	BPEL 1.1		
SKUPINA VZORCEV	**VSI VZORCI**	**PODPRTI VZORCI**	**% podprtih**
Osnovni vzorci	5	5	100,00 %
Napredni vzorci z razcepi in sinhronizacijami	14	5	35,71 %
Vzorci s podporo več instancam aktivnosti	7	1	14,29 %
Vzorci za upravljanje s stanji	5	4	80,00 %
Preklicni vzorci	5	3	60,00 %
Ponovitveni vzorci	3	1	33,33 %
Zaustavitveni vzorci	2	1	50,00 %
Sprožilni vzorci	2	1	50,00 %
SKUPAJ	**43**	**21**	**48,84 %**

Vir: Prirejeno po N. Russel et al., Workflow Control-Flow Patterns: A Revised View, 2006, str. 79 in 81.

Tabela 7: Podpora skupinam vzorcev BPEL 2.0

	BPEL 2.0		
SKUPINA VZORCEV	**VSI VZORCI**	**PODPRTI VZORCI**	**% podprtih**
Osnovni vzorci	5	5	100,00 %
Napredni vzorci z razcepi in sinhronizacijami	14	3	21,43 %
Vzorci s podporo več instancam aktivnosti	7	5	71,43 %
Vzorci za upravljanje s stanji	5	5	100,00 %
Preklicni vzorci	5	5	100,00 %
Ponovitveni vzorci	3	1	33,33 %
Zaustavitveni vzorci	2	2	100,00 %
Sprožilni vzorci	2	1	50,00 %
SKUPAJ	**43**	**27**	**62,79 %**

Vir: Prirejeno po J. Lenhard, A Pattern-based Analysis of WS-BPEL and Windows Workflow, 2011, str. 77.

Raziskava, ki jo je opravil Lenhard (2011), prikazuje (Tabela 7) velik vpliv nadgradnje predvsem v skupini »Vzorci s podporo več instancam«, kjer so z uvedbo možnosti vzporednega izvajanja z aktivnostjo »forEach« omogočili podporo 71,43 odstotka vzorcev, v primerjavi z 14,29-odstotno podporo, ki jo je nudila tehnologija BPEL 1.1. Poudariti pa je potrebno tudi slabšo podporo predvsem vzorcem z razcepi in sinhronizacijami (skupina 2), kjer pride do izraza pomanjkanje podpore izvajanju večnitnih instanc ene veje procesa, slabosti pa se izražajo tudi zaradi visoke strukturiranosti tehnologije in težav pri izvedbi cikličnih poslovnih procesov takrat, ko je potrebno preverjanje statusa predhodnih aktivnosti ob sinhronizaciji vej. Kot je prikazano v Tabeli 7, BPEL 2.0 skupaj **podpira 62,79 odstotka vzorcev** delovnega toka, kar v primerjavi s predhodno različico tehnologije pomeni **izboljšavo za 13,95 odstotne točke**.

Tabela 8: Podpora skupinam vzorcev Workflow Foundation 3.5

	WF 3.5		
SKUPINA VZORCEV	**VSI VZORCI**	**PODPRTI VZORCI**	**% podprtih**
Osnovni vzorci	**5**	5	100,00 %
Napredni vzorci z razcepi in sinhronizacijami	**14**	7	50,00 %
Vzorci s podporo več instancam aktivnosti	**7**	5	71,43 %
Vzorci za upravljanje s stanji	**5**	5	100,00 %
Preklicni vzorci	**5**	4	80,00 %
Ponovitveni vzorci	**3**	2	66,67 %
Zaustavitveni vzorci	**2**	1	50,00 %
Sprožilni vzorci	**2**	2	100,00 %
SKUPAJ	**43**	**31**	**72,09 %**

Vir: Prirejeno po M. Zapletal et al., An Analysis of Windows Workflow's Control-Flow Expressiveness, 2009, str. 200–209.

Rezultati raziskave podpore Windows Workflow Foundation 3.5 vzorcem delovnega toka (Tabela 8) prikazujejo nekoliko slabšo podporo večnitnim procesnim modelom, ki se izvajajo v isti instanci delovnega toka, saj Workflow Foundation podpira izvajanje le ene niti istočasno. To je ključen razlog, da ima zelo nizek odstotek podpore ravno skupina »Napredni vzorci z razcepi in sinhronizacijami«. Prav tako enako velja za poslovne procese modelirane z modelom avtomata stanj, kjer lahko instanca ohranja le eno stanje v določenem trenutku. Avtorji raziskave trdijo, da bi v primeru podpore večnitnemu izvajanju tehnologija podpirala večino od 43 vzorcev delovnega toka. Podpora vzorcem delovnega toka tehnologije Windows Workflow Foundation 3.5 je 72,09-odstotna, saj podpira 31 od 43 vzorcev. Dejstvo pa je, da je takšen rezultat dosežen z uporabo obeh modelov razvoja poslovnih programskih rešitev, ki jih tehnologija podpira

(zaporedni in avtomat stanj). Avtomat stanj je namreč posebej primeren za upravljanje delovnih tokov, kjer je izrazito prisotna interakcija s človeškim uporabnikom programske rešitve.

Novejša različica tehnologije Workflow Foundation, 4.0 po raziskavi Lenharda (2011), kot je razvidno iz Tabele 9, podpira en vzorec manj od predhodne različice. Kot je že v opisu tehnologije omenjeno, je med različicama kar veliko sprememb. Kljub temu pa večjega napredka pri podpori vzorcem delovnih tokov ni bilo, v skupini naprednih vzorcev z razcepi je podpora celo nižja, podprt je namreč en vzorec manj v primerjavi z različico 3.5. Ključni razlog za takšen rezultat je umik avtomata stanj, ki je bil v predhodni različici prisoten. Ta je omogočal enostavnejše upravljanje delovnih tokov, ki so zahtevali upravljanje stanj delovnega toka. Dodatna pomanjkljivost tehnologije, ki je iz raziskave razvidna pa je slaba podpora nestrukturiranim delovnim tokovom, saj ne podpira vseh zmogljivosti, ki jih omogoča razvoj programskih rešitev z diagramom toka podatkov. Takšen primer so veje delovnih tokov, ki se izvajajo istočasno. Workflow Foundation 4.0 **podpira 69,77 odstotka** vzorcev delovnih tokov, **30 od 43 vzorcev**. Podpora v primerjavi s predhodno različico se je **znižala za 2,32 odstotne točke.** Omeniti je potrebno, da je Microsoft z nadgradnjo tehnologije na različico 4.5 tudi ponovno omogočil uporabo avtomata stanj, kar danes posledično lahko pomeni višjo podporo nekaterim vzorcem v primerjavi z rezultati obravnavane študije.

Tabela 9: Podpora skupinam vzorcev Workflow Foundation 4.0

	WF 4.0		
SKUPINA VZORCEV	**VSI VZORCI**	**PODPRTI VZORCI**	**% podprtih**
Osnovni vzorci	5	5	100,00 %
Napredni vzorci z razcepi in sinhronizacijami	14	6	42,86 %
Vzorci s podporo več instancam aktivnosti	7	5	71,43 %
Vzorci za upravljanje s stanji	5	5	100,00 %
Preklicni vzorci	5	5	100,00 %
Ponovitveni vzorci	3	1	33,33 %
Zaustavitveni vzorci	2	2	100,00 %
Sprožilni vzorci	2	1	50,00 %
SKUPAJ	**43**	**30**	**69,77 %**

Vir: Prirejeno po J. Lenhard, A Pattern-based Analysis of WS-BPEL and Windows Workflow, 2011, str. 77.

V preseku sta tehnologiji BPEL 2.0 in Workflow Foundation 4.0 zelo podobni. Workflow Foundation sicer v primerjavi s konkurenčno tehnologijo podpira skoraj vse diskriminacijske vzorce in veliko bolje upravlja s preklici izvajanja aktivnosti, na drugi strani pa ima večje težave z obvladovanjem nestrukturiranih poslovnih procesov. Obema tehnologijama pa je skupno to, da

je večnitno izvajanje instanc istega delovnega toka nemogoče. Microsoftova tehnologija sicer v odstotkih podpira več vzorcev delovnih tokov (6,98 odstotne točke več), iz tega lahko sklepamo, da bi v primeru, če bi odločitev o uporabi ene ali druge tehnologije temeljila na vzorcih delovnih tokov, bi uporabniki izbirali Microsoftov Workflow Foundation.

4.3 Sinteza primerjave

Microsoftovo razvojno ogrodje .NET v povezavi z lastno tehnologijo Windows Workflow Foundation (WF) in razvojna platforma Java v povezavi z odprtokodnim standardom Business Process Execution Language (BPEL) sta v zadnjem desetletju prevladujoči in najmnožičneje uporabljeni tehnologiji na področju razvoja in izvajanja poslovnih programskih rešitev, ki temeljijo na delovnih tokovih. Na podlagi zadnjih raziskav sta sicer tehnologiji med uporabniki zastopani v približno enakem razmerju (vsaka ima 40-odstotni delež). Vsaka izmed tehnologij sicer temelji in se izvaja na različnem razvojnem ogrodju, kar je tudi lahko razlog za tak rezultat, saj izbira tehnologije za izvajanje in razvoj poslovnih programskih rešitev, ki temeljijo na delovnih tokovih, ni nujno edini razlog za takšno odločitev, pogosto odločitev o uporabljeni tehnologiji narekuje izbira enega ali drugega razvojnega ogrodja.

Na tržišče je prvi prispel BPEL, tehnologijo je razvila skupina, sestavljena iz vodilnih podjetij, ki ponujajo razvojna orodja za gradnjo poslovnih programskih rešitev, Microsoft, IBM, SAP, BEA Systems in Siebel Systems. Cilj razvoja BPEL je bil razvoj tehnologije, ki bi omogočala izvajanje in razvoj poslovnih programskih rešitev, ki temeljijo na delovnih tokovih, vendar pa so vodilni proizvajalci orodij želeli tudi obvladovati vse pomembne jedrne tehnologije tistega časa. V obravnavanem obdobju je namreč zaživela tudi paradigma SOA (storitveno orientirana arhitektura) skupaj s spletnimi storitvami, ki je nakazovala smernice razvoja jedrnih tehnologij v prihodnosti. Z velikimi vložki, tudi v trženjske kampanje, so tako razvili svojo tehnologijo ter izrinili ostale obstoječe tehnologije. Dejstvo pa je, da pregled tehnologije BPEL in ostalih takratnih konkurenčnih tehnologij prikazuje še dodaten zorni kot, to pa je sama zgradba. BPEL je namreč bil razvit tako, da omogoča integracijo z razvojnimi ogrodji in za delovanje potrebuje razvojno ogrodje, medtem ko takrat konkurenčne tehnologije tega niso potrebovale, lahko so nastopale samostojno. S strani proizvajalcev orodij je to predstavljalo prej slabost kot prednost, saj je za njih bilo in še vedno je ključnega pomena to, da različne tehnologije integrirajo v svoja razvojna ogrodja, celovite rešitve, saj moderna poslovna programska rešitev potrebuje še mnogo drugih zmogljivosti za izpolnitev poslovnih zahtev, ne le zmožnost izvajanja na podlagi delovnega toka.

Podjetje Microsoft poleg jedrnih tehnologij razvija in trži tudi večini dobro poznane tehnologije in rešitve, kot je operacijski sistem Microsoft Windows, zbirko pisarniških programskih rešitev Microsoft Office, strežniške tehnologije ipd. Skratka s svojimi tehnologijami in orodji pokriva večji spekter informacijskih tehnologij ter večino uporabnikov računalnikov po svetu. Ko so opazili, da se tehnologije kot je BPEL množično uporabljajo pri razvoju poslovnih programskih rešitev, so se odločili za razvoj lastne tehnologije. Odločitev je po našem mnenju smiselna. Sami že imajo lasten operacijski sistem, ki je v množični uporabi (MS WINDOWS), orodja za

pisarniško delo (MS Office), razvojno ogrodje (.NET) s skupkom pripadajočih standardov. Podjetje se je odločilo unovčiti sinergije, ki jih lahko doseže z integracijo podpore delovnim tokovom v svoje obstoječe produkte, hkrati pa lahko tudi unovči znanje in izkušnje, ki izhajajo iz odzivov strokovne javnosti na uporabnost in pomanjkljivosti tehnologije BPEL. Razvili so Windows Workflow Foundation (WF), ki je, kot pravijo, »izvajalnik delovnih tokov, programski model in nabor orodij za razvijalce«, BPEL-tehnologijo pa podpirajo do te mere, da jim v večini primerov služi le kot orodje za izvoz/uvoz in prenos delovnih tokov med razvojnimi ogrodji.

V primerjalni analizi podpore vzorcev delovnih tokov z analizo treh ločenih raziskav, in sicer:

- Russel et al. iz leta 2006 – BPEL 1.1.;
- Zapletal et al. iz leta 2009 – Workflow Foundation 3.5;
- Lenhard iz leta 2011 – BPEL 2.0 in Workflow Foundation 4.0;

smo preverili, kakšne so dejanske razlike med tehnologijami v praksi. 43 vzorcev delovnih tokov služi kot merilo za evalvacijo možnosti izražanja posameznih programskih jezikov v praksi, vzorci pa služijo tudi kot strokovna primerjava med posameznimi tehnologijami.

Primerjave so bile narejene na dveh ločenih različicah posamezne tehnologije, na podlagi katerih je razviden tudi razvoj izboljšav. BPEL 1.1. je na podlagi raziskave podpiral 48,48 odstotka vseh vzorcev, z večjimi pomanjkljivostmi pri podpori vzorcev skupin »Napredni vzorci z razcepi in sinhronizacijami« (35,71 odstotka), »Vzorci s podporo več instancam aktivnosti« (14,29 odstotka) in »Ponovitveni vzorci« (33,33 odstotka). V istem obdobju je takrat nova Microsoftova tehnologija podpirala 72,09 odstotka vseh vzorcev, z najslabšo podporo v skupinah »Napredni vzorci z razcepi in sinhronizacijami« (50 odstotkov) in »Zaustavitveni vzorci« (50 odstotkov).

S prehodom na novejši različici je največji napredek viden pri tehnologiji BPEL, kjer je podpora narasla za 13,95 odstotne točke na 62,79 odstotka po večini zaradi implementacije vzporednega izvajanja, še vedno pa ostaja šibka točka, skupina »Napredni vzorci z razcepi in sinhronizacijami«, kjer se odraža pomanjkanje podpore večnitnim instancam veje procesa. Drugačna slika se kaže pri študiji podpore vzorcev tehnologije WF 4.0, kjer se je podpora znižala za 2,32 odstotne točke na 69,77 odstotka. Razlog za upad podpore je v umiku avtomata stanj kot načina razvoja in izvajanja delovnih tokov, kar pa je bilo v novi nadgradnji 4.5. odpravljeno. Pričakovati je, da je trenutno podpora vzorcem nekoliko višja, kot to prikazuje obravnavana študija primera.

Tehnologiji sta na podlagi podpore vzorcev delovnih tokov zelo primerljivi, BPEL ima sicer več težav pri podpori diskriminacijskih vzorcev, medtem kot se WF slabše znajde s podporo nestrukturiranih poslovnih procesov, obema pa je skupno to, da imata težave pri podpori večnitnim instancam posameznih vej poslovnih procesov. Če bi se za odločitev o izbiri ene tehnologije v primerjavi z drugo odločali na podlagi vzorcev delovnih tokov, bi se odločili za tehnologijo Workflow Foundation, saj v primerjavi s tehnologijo BPEL podpira 6,98 odstotne točke več vzorcev.

SKLEP

Kot smo že v uvodu omenili, smo v magistrski nalogi preučili, katere so sodobne jedrne tehnologije, ki omogočajo razvoj poslovnih programskih rešitev na podlagi poslovnih procesov posamezne organizacije. Osredotočili smo se torej na fazo informatizacije poslovanja in poslovnih procesov ter še konkretneje na jedrne razvojne tehnologije in razvojna orodja, ki se pri tem najmnožičneje uporabljajo; zanimalo nas je, kakšne so bile smernice razvoja tovrstnih tehnologij od časa, ko so postale predmet množične uporabe, na koncu pa smo pripravili primerjalno analizo že obstoječih raziskav med dvema najmnožičneje uporabljenima tehnologijama.

Na podlagi Gartnerjevih študij smo razbrali, da orodja v povezavi z javansko platformo temeljijo na tehnologiji BPEL oz. na tehnologiji WF v povezavi z Microsoftovo platformo .NET. V uporabljeni raziskavi sicer ni izrecno poudarjeno, da je BPEL-tehnologija, na kateri dotična orodja temeljijo, vendar pa je na podlagi več strokovnih gradiv mogoče razbrati, da je BPEL t. i. *de-facto* industrijski standard na področju izvajanja delovnih tokov, sam BPEL pa deluje v kombinaciji z javansko platformo.

Ugotovili smo, da so smernice razvoja tehnologij večkrat spremenile smer, vodilna podjetja, ki na trgu nudijo razvojna orodja za razvoj poslovnih programskih rešitev, so namreč želela imeti nadzor nad jedrnimi tehnologijami in tako oblikovati strategije prihodnjih tehnologij. Kljub začetnim skupnim vložkom se je vodilno podjetje Microsoft naknadno odločilo za svojo lastno strategijo ter tako trgu in razvijalcem predstavilo svojo jedrno tehnologijo Workflow Foundation, ki je nadomestek do tedaj obstoječi tehnologiji Business Process Execution Language. Do tega je prišlo, ker je Microsoft med drugim želel izkoristiti sinergije in konkurenčno prednost, ki jo ima zaradi široke palete lastnih tehnologij in množične uporabe le-teh.

Iz tega naslova so se pojavila nam zanimiva vprašanja: ali je Microsoftova tehnologija zelo različna od BPEL, ali obstajajo večje razlike med tehnologijama in katera je boljša. Na podlagi analize teoretičnih virov smo ugotovili, da obstaja znanstvena metodologija primerjave tehnologij, ki so namenjene izvajanju in razvoju poslovnih programskih rešitev, ki temeljijo na delovnih tokovih. Metodologija vsebuje 43 vzorcev delovnih tokov ter omogoča evalvacijo tehnologij, katere omogočajo izvajanje delovnih tokov.

Med seboj smo primerjali že obstoječe študije, ki so bile izvedene na dveh različicah vsake obravnavane jedrne tehnologije (na prvotni in aktualni izboljšani različici). Iz primerjalne analize študij izhaja, da so jedrne tehnologije med seboj primerljive v obsegu, ki ga metodologija zajema, obstajajo pa razlike med njima. Posebej so razlike vidne med aktualnimi in prvotnimi različicami, viden je napredek v smer podpore večini situacij, ki so običajne pri razvoju poslovnih programskih rešitev. Iz primerjalne analize izhaja, da bi, če bi odločitev o izbiri ene tehnologije v primerjavi z drugo temeljila na vzorcih delovnih tokov, izbrali tehnologijo WF. To pa ne pomeni nujno, da je ta izbira optimalna, saj omenjene tehnologije temeljijo na različnih

razvojnih ogrodjih, kar je eden izmed dejavnikov, ki tudi vplivajo na izbiro, na odločitev pa lahko vpliva tudi opravljena primerjalna analiza.

LITERATURA IN VIRI

1. Aboulnaga, A., & Pareek, A. (2012). *Oracle Soa Suite 11G Administrator's Handbook.* Birmingham: Pact Publishing Ltd.
2. *Activities Overview*. Najdeno 8. marca 2016 na spletnem naslovu https://msdn.microsoft.com/en-us/library/aa349008(v=vs.90).aspx
3. Alexander, C. (1978). *A Pattern Language.* Oxford: Oxford University Press.
4. *Application Layers*. Najdeno 8. marca 2016 na spletnem naslovu https://help.sap.com/saphelp_nwce711/helpdata/en/7e/d1a40b5bc84868b1606ce0dc72d88b/content.htm
5. Behara, G. K. (2006). BPM and SOA: A Strategic Alliance. *BPTrends*. Najdeno 8. marca 2016 na spletnem naslovu http://www.bptrends.com/publicationfiles/05-06-WP-BPM-SOA-Behara.pdf
6. Benedik, M. (2013) *Modeliranje in izvajanje poslovnih procesov z uporabo BPMN 2.0* (diplomsko delo). Ljubljana: Fakulteta za računalništvo in informatiko.
7. Blow, M., Goland, Y., Kloppmann, M., Leymann, F., Pfau, G., Roller, D., & Rowley, M. (2004). *BPELJ: BPEL for Java*. Arlington: BEA Systems.
8. Booch, G. (2006). The Accidental Architecture. *IEEE Software, 23*(3), 9-11.
9. Božnik, J. (2010). *Razvoj delovnih tokov in aplikacij za platformo Sharepoint* (diplomsko delo). Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko.
10. *BPM Need V/S BPEL Oracle 11g*. Najdeno 8. marca 2016 na spletnem naslovu https://community.oracle.com/message/11346786#11346786
11. BPMN 2.0 vs BPEL - Oracle BPM VS SOA Suite. *M&S Consulting*. Najdeno 8. marca 2016 na spletnem naslovu http://www.mandsconsulting.com/bpmn-20-vs-bpel-oracle-bpm-vs-soa-suite
12. *BPMN vs. BPEL*. Najdeno 8. marca 2016 na spletnem naslovu https://www.ibm.com/developerworks/community/forums/html/topic?id=e46b13d0-6342-4b63-bfe0-325b421f61c1
13. Campione, M., Walrath, K., & Huml, A. (2001). *The Java Tutorial - A Short Course on the Basics (3rd ed.)*. Boston: Addison - Wesley.
14. Carr, N. G. (2003). IT Doesn't Matter. *Harvard Business Review, 81*(5), 41-49.
15. Chappell, D. (2007a). *Microsoft and BPM: A Technology Overview*. Redmond: Microsoft Corporation.
16. Chappell, D. (2007b, 28. februar). Windows Workflow Foundation and BPEL. *David Chappell Opinari.* Najdeno 8. marca 2016 na spletnem naslovu http://davidchappellopinari.blogspot.si/2007/02/windows-workflow-foundation-and-bpel.html
17. Collins, M. (2010). *Beginning WF - Windows Workflow in .NET 4.0*. New York: Apress.
18. Coupelon, O. (2007). *Analyse et optimisation de l'architecture des moeurs BPEL* (magistrsko delo). Aubiere: LIMOS.
19. Cunha, P. F., & Maropoulos, P. G. (2007). *Digital Enterprise Technology*. Berlin: Springer.
20. Davenport, T. (1993). *Process Innovation: Reengineering Work through Information Technology*. Boston: Harward Business School Press.

21. De Hullu, C. (2012). *Evaluating .NET-Based Enterprise Service Bus Solutions.* Twente: University of Twente.
22. Delbecq, A. L., & Van de Ven, A. H. (1971). A Group Process Model for Problem Identification and Program Planning. *Journal of Applied Behavioral Science 7*(4) 466 – 492.
23. *Developing Applications with the Workflow Designer*. Najdeno 8. marca 2016 na spletnem naslovu https://msdn.microsoft.com/en-us/library/dd489396.aspx
24. Duine, R., Shulte, W. R., Cantara, M., & Kerremans, M. (2015). *Magic Quadrant for Intelligent Business Process Management Suites*. Stanford: Gartner.
25. Dumas, M., van der Aalst, W., & ter Hofstede, A. (2005). *Process-Aware Information Systems.* Hoboken: John Wiley & Sons.
26. Ellis, C. A. (1999). Workflow technology. V M. Beaudouin-Lafon (ur.), *Computer Supported Co-operative Work*, (str. 29-54). Hoboken: John Wiley & Sons.
27. Fengel, J. (2013). Business Semantics Alignment for Business Process Model Integration. V K. Tarnay, L. Xu & S. Imre (ur.), *Research and Development in E-Business through Service-Oriented Solutions*, (str. 91-112). Hershey: IGI Global.
28. Fischer, L. (2004). *Workflow Handbook*. Florida: Future Strategies Inc.
29. Fleming, M., & Silverstein, J. (2011). Worldwide Business Process Platforms 2011 Vendor Analysis. *IDC*. Najdeno 10. junija 2015 na spletnem naslovu http://www.idevnews.com/images/emailers/1112231_Progress_KathyO_75k/IDC%202011%20MarketScape%20BP%20Platforms.pdf."
30. Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Boston: Addison Wesley
31. Freundstein, P. (2009). *Web Engineering for Workflow-based Applications: Models, Systems and Methodologies.* Karlsruhe: Universitatsverlag Karlsruhe.
32. Freundstein, P., Buck, J. Nussbaumer, M., & Gaedke, M. (2007). Model-driven Construction of Workflow-based Web Applications with Domain-specific Languages. *Proceedings of the 3rd international workshop MDWE2007* (str. 1–15). Munchen: Ludwig-Maximilian University.
33. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software.* Boston: Addison-Wesley Longman Publishing.
34. Gradišar, M., Jaklič, J., Damij, T., & Baloh, P. (2005). *Osnove poslovne informatike.* Ljubljana: Ekonomska fakulteta.
35. HandySoft Corporation. (2003). *Business Process Management and its Value to the Enterprise.* Virginia: HandySoft Corporation.
36. Harmon, P. (2003). *Business Process Change: A Manager's Guide to Improving, Redesigning, and Automating Processes*. San Francisco: Morgan Kaufmann Publishers.
37. Hill. J. B., Cantara, M., Kerremans, M., & Plummer, D. C. (2009). *Magic Quadrant for Business Process Management Suites.* Stamford: Gartner.
38. IBM Redbooks. (2013). *WebSphere Application Server V8.5 Concepts, Planning, and Design Guide.* New York: IBM Redbooks.
39. *InfoQ Interviews BPEL4People Representatives*. Najdeno 21. januarja 2016 na spletnem naslovu http://www.infoq.com/articles/bpel4people-tc
40. Ionita, A. D., & Estublier, J. (2010). Business Process Modeling and Automation with General and Domain Specific Languages. V J. A. Beckmann (ur.), *Business Process*

Modeling : Software Engineering, Analysis and Applications (str. 63–94). Hauppauge: Nova Science Publishers.

41. Jellema, L. (2011). Build Process-Oriented Applications. *Oracle*. Najdeno 10. aprila 2015 na spletnem naslovu http://www.oracle.com/technetwork/issue-archive/2011/11-may/o31bpm-354086.html
42. Jendrock, E., Cervera-Navarro, R., Evans, I., Gollapudi, D., Haase, K., Markito, W., & Srivathsa, C. (2013). The Java EE 6 Tutorial. *Oracle*. Najdeno 8. marca 2016 na spletnem naslovu http://docs.oracle.com/javaee/6/tutorial/doc/docinfo.html
43. Jurič, M. B., & Weerasiri, D. (2014). *WS-BPEL 2.0 Beginner's Guide*. Birmingham: Pact Publishing Ltd.
44. Jurič, M. B. (2005). Storitvena arhitektura - zgolj kompozicija spletnih storitev? *SOA Competence Center*. Najdeno 8. marca 2016 na spletnem naslovu http://www.soa.si/juric/soa_ss.pdf
45. Jurič, M. B., & Pant, K. (2008). *Business Process Driven SOA using BPMN and BPEL*. Birmingham: Packt Publishing.
46. Jurišić, M. (2011). Transition between process models (BPMN) and service models (WS-BPEL and other standards): A systematic review. *Journal of Information and Organizational Sciences, 35*(2), 163 - 171.
47. Khalaf, R., Mukhi, N., Curbera, F., & Weerawarana, S. (2005). The Business Process Execution Language. V M. Dumas, W. van der Aalst & A. ter Hofstede (ur.), *Process-Aware Information Systems* (str. 317 – 342). Hoboken: John Wiley & Sons.
48. Khan, R. N. (2004). *Business Process Management. A Practical Guide*. Tampa: Meghan-Kiffer Press.
49. Kloppmann, M., Koenig, D., Leymann, F., Pfau, G., Rickayzen, A., von Riegen, C., Schmidt, P., & Trickovic, I. (2005). WS-BPEL Extension for Sub-processes - BPEL-SPE. *IBM in SAP*. Najdeno 8. marca 2016 na spletnem naslovu http://xml.coverpages.org/BPEL-SPE-Subprocesses.pdf
50. Ko, R. K. L., Lee, S. S. G., & Lee, E. W. (2009). Business process management (BPM) standards: a survey. *Business Process Management Journal 15*(5) 744-791.
51. Kovačič, A., & Bosilj Vukšić, V. (2005). *Management poslovnih procesov: Prenova in informatizacija poslovanja*. Ljubljana: GV Založba.
52. Kovačič, A., Jaklič, J., Indihar Štemberger, M., & Groznik, A. (2004). *Prenova in informatizacija poslovanja*. Ljubljana: Ekonomska fakulteta.
53. Kurbel, E. K. (2008). *The Making of Information Systems*. Berlin: Springer-Verlag.
54. Laudon, K. C., & Laudon, J. P. (2007). *Management Information Systems: Managing the Digital Firm (10th ed.)*. Upper Saddle River: Pearson Education.
55. Lenhard, J. (2011). *A Pattern-based Analysis of WS-BPEL and Windows Workflow*. Bamberg: Distributed Systems Group.
56. Lind, M. (2005). Contextual Understanding of Information Systems - Characteristics of Process Oriented Information Systems. *Proceedings of the Sixteenth Australasian Conference on Information Systems (ACIS 2005)* (str. 45-55). Sydney: University of Technology.

57. Little, M. (2008, 24. junij). InfoQ Interviews BPEL4People Representatives. *InfoQ*. Najdeno 8. marca 2016 na spletnem naslovu http://www.infoq.com/articles/bpel4people-tc
58. Liu, H. H. (2011). *Software Performance and Scalability*. Hoboken: Johny Wiley & Sons, Inc.
59. Manouvrier, B., & Menard, L. (2008). *Application Integration: EAI, B2B, BPM and SOA.* Hoboken: John Wiley & Sons, Inc.
60. Marolt, J., & Gomišček, B. (2005). *Management kakovosti.* Kranj: Moderna organizacija.
61. McKendrick, J. (2007, 1. marec). Microsoft applies BPEL to Windows Workflow; analysts scratch their heads. *Zdnet.* Najdeno 8. marca 2016 na spletnem naslovu http://www.zdnet.com/article/microsoft-applies-bpel-to-windows-workflow-analysts-scratch-their-heads/
62. *Microsoft Redefines Workflow With Windows Workflow Foundation*. Najdeno 8. marca 2016 na spletnem naslovu http://news.microsoft.com/2005/09/14/microsoft-redefines-workflow-with-windows-workflow-foundation/
63. Myers, B. R. (2007). Windows Workflow Foundation and Web Services. Dr. Dobb's Journal. *Dr. Dobb's. The world of Software Development.* Najdeno 10.6.2015 na spletnem naslovu http://www.drdobbs.com/windows/windows-workflow-foundation-and-web-serv/198000882
64. *N-Tier Data Applications Overview*. Najdeno 8. marca 2016 na spletnem naslovu https://msdn.microsoft.com/en-us/library/bb384398.aspx
65. *OASIS Web Services Business Process Execution Language (WSBPEL) TC*. Najdeno 8. marca 2016 na spletnem naslovu https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
66. Oasis. (2007). Web Services Business Process Execution Language Version 2.0. *Oasis*. Najdeno 8. marca 2016 na spletnem naslovu http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf
67. Oesterle, H. (1995). *Business in the Information Age: Heading for new processes.* Berlin: Springer.
68. Oracle. (2006). Interoperating J2EE and Microsoft .Net Applications - What Standards to Adopt? *Oracle*. Najdeno 8. marca 2016 na spletnem naslovu http://www.oracle.com/technetwork/database/windows/j2eedotnet-1-131924.pdf
69. Oracle. (b.l.). Business Process Management and WS-BPEL 2.0. *Oracle*. Najdeno 8. marca 2016 na spletnem naslovu http://www.oracle.com/technetwork/topics/bpel-130653.pdf
70. Ouyang, C., van der Aalst, W., Dumas, M., & ter Hofstede, A. H. M. (2006). From Business Process Models to Process-oriented Software Systems. *ACM Transactions on Software Engineering and Methodology*, *19*(1), 41–77.
71. *Overview of the .NET Framework.* Najdeno 8. marca 2016 na spletnem naslovu https://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.110).aspx
72. *Persistence Overview*. Najdeno 8. marca 2016 na spletnem naslovu https://msdn.microsoft.com/en-us/library/aa349367(v=vs.90).aspx
73. Petriu, D., Rouquette, N., & Haugen, O. (2010). *Model Diven Engineering Languages and Systems.* Berlin: Springer.
74. *Port Workflow Foundation to CoreFx/CoreCLR*. Najdeno 8. marca 2016 na spletnem naslovu https://github.com/dotnet/corefx/issues/2394

75. Preac, A. (2011). *Uporaba BPMN za modeliranje upravnih postopkov* (diplomsko delo). Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko.
76. Ribić, S. (2015). *Model pretvorbe BPEL v Amazon Simple Workflow Service* (magistrsko delo). Ljubljana: Fakulteta za računalništvo in informatiko.
77. Rozman, T. (2010). Upravljanje poslovnih procesov v podjetjih: principi, metode in človeški faktorji (Nastopno predavanje za izvolitev v naziv docent). Najdeno 20. julija 2015 na spletnem naslovu http://www.slideshare.net/tomirozman/upravljanje-poslovnih-procesov-v-podjetjih-principi-metode-in-loveki-faktorji-nastopno-predavanje-doba-fakulteta.
78. Russel, N., ter Hofstede, A. H. M., Edmond, D., & van der Aalst, W. M .P. (2005). Workflow Data Patterns: Identification, Representation and Tool Support. V L. Delcambre, C. Kop, H. C. Mayr, J. Mylopoulos & O. Pastor (ur.), *Proceedings of the 24th International Conference on Conceptual Modeling (ER 2005)* (str. 353-368). Berlin: Springer - Verlag.
79. Russel, N., ter Hofstede, A. H. M., van der Aalst, W.M.P., & Mulyar, N. (2006). Workflow Control-Flow Patterns: A Revised View. *BPM Center*. Najdeno 8. marca 2016 na spletnem mestu http://bpmcenter.org/wp-content/uploads/reports/2006/BPM-06-22.pdf
80. Saraswathi, R., & Singh, J. (2013). *Oracle SOA BPEL Process Manager 11gR1 - A Hands-on Tutorial*. Birmingham: Packt Publishing.
81. *Serialization Overview*. Najdeno 8. marca 2016 na spletnem naslovu https://msdn.microsoft.com/en-us/library/aa349369(v=vs.90).aspx
82. Seroter, R., Fairweather, E., & Ramani, R. (2010). *Applied Architecture Patterns on the Microsoft Platform: An In-depth, Scenario-driven Approach to Architecting Systems Using Microsoft Technologies*. Birmingham: Pact Publishing Ltd.
83. Sharp, A., & McDermott, P. (2001). *Workflow modelling. Tools for process improvement and application development*. Norwood: Artech House Inc.
84. Sharp, J. (2010). *Windows Communication Foundation 4 Step by Step*. Redmond: Microsoft Corporation.
85. Sikos, L. (2015). *Mastering Structured Data on the Semantic Web*. New York: Apress.
86. Singer, R., Kotremba, J., & Raß, S. (2014), Modeling and Execution of Multienterprise Business Processes. *Proceedings CBI 2014: 16th IEEE Conference on Business Informatics* (str. 68-73). Ženeva: IEEE.
87. Sinur, J., & Hill J. B. (2010). *Magic Quadrant for Business Process Management Suites.* Stamford: Gartner
88. Smith, H. (2005). And Your Future BPMS Is? Microsoft Office. *BPTrends*. Najdeno 8. marca 2016 na spletnem naslovu http://www.bptrends.com/publicationfiles/12-05-ART-YourFutureBPMS-Office-Smith.pdf
89. Smith, H. (2008). BPMS 2008 - Look Back to Look Forward. *BPTrends.* Najdeno 8. marca 2016 na spletnem naslovu http://www.bptrends.com/publicationfiles/ONE%2002-08-ART-BPMS-2008-LookBack2LookForward-Smith-doc%20V03.pdf
90. Smith, H., & Fingar, P. (2003). Business Process Management: The Third Wave. *BPTrends.* Najdeno 8. marca 2016 na spletnem naslovu http://www.bptrends.com/publicationfiles/BPM%20Third%20Wave%20Smith%20Fingar%20Apr2003.pdf

91. Smith, W. H. & Gray, C. R. (2010). *Teaching/Learning Strategies for Technology Education.* Maryland: Technology Education Association Maryland.
92. Strupe, M. K. (2010, 23. junij). Workflow Automation - Rembembering Where BPM Came From. *Ultimus Enterprise Solutions*. Najdeno 8. marca 2016 na spletnem naslovu http://www.ultimus.com/Blog/bid/46197/Workflow-Automation-Remembering-Where-BPM-Came-From
93. Ter Hofstede, A. H. M., van der Aalst, W., Adams, M., & Russell, N. (2010). *Modern Business Process Automation. Yawl and its Support Environment.* Berlin: Springer-Verlag.
94. Bass, L., Clements, P., & Kazman, R. (2003). *Software Architecture in Practice. Second Edition*. Boston: Addison - Wesley.
95. Terceros, C. A., & Leslie, S. (2011). *Oracle Fusion Middleware.* Redwood City: Oracle America, Inc.
96. The Yawl Foundation. (2012). YAWL – User Manual, version 2.3.2012. *The Yawl Foundation.* Najdeno 8. marca 2016 na spletnem naslovu http://yawlfoundation.org/manuals/YAWLUserManual2.3.pdf
97. Van der Aalst, W. M. P., Barros, A. P., ter Hofstede, A. H. M., & Kiepuszewski, B. (2000). Advanced Workflow Patterns. V O. Etzion & P. Scheuermann (ur.), *Cooperative Information Systems* (str. 18 - 29). Berlin: Springer.
98. Van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., & Barros, A. P. (2003). Workflow Patterns. *Distributed and Parallel Databases, 14*(1), 5–51.
99. *Wanted: Enterprise Architects.* Najdeno 8. marca 2016 na spletnem naslovu http://scn.sap.com/docs/DOC-17716
100. Weske, M. (2012). *Business Process Management Concepts, Languages, Architectures (2nd ed.)*. Berlin: Springer.
101. White, B. (2013). *Pro WF 4.5*. New York: Apress.
102. *Windows Workflow Foundation Overview*. Najdeno 8. marca 2016 na spletnem naslovu https://msdn.microsoft.com/en-us/library/bb628449.aspx
103. *Windows Workflow Tutorial*. Najdeno 8. marca 2016 na spletnem naslovu https://msdn.microsoft.com/en-us/library/dd692929.aspx
104. *Workflow and Process.*. Najdeno 8. marca 2016 na spletnem naslovu https://msdn.microsoft.com/en-us/library/bb833024.aspx
105. *Workflow in the Future.* Najdeno 20. februarja 2016 na spletnem naslovu https://social.msdn.microsoft.com/Forums/vstudio/en-US/1bdc2a53-b9f4-4799-aab0-eb158f81ec3e/workflow-in-the-future?forum=wfprerelease
106. Workflow Management Coalition. (1996). *The Workflow Management Coalition Specification. Terminology & Glossary*. Bruselj: Workflow Management Coalition.
107. *Workflow Markup Overview*. Najdeno 8. marca 2016 na spletnem naslovu https://msdn.microsoft.com/en-us/library/aa349003(v=vs.90).aspx
108. Xu, L., & de Vrieze, P. (2013). Service Oriented Business Process Modeling Today and Tomorow. V K. Tarnay, L. Xu & S. Imre (ur.), *Research and Development in E-Business through Service-Oriented Solutions* (str. 113-133). Hershey: IGI Global.
109. Xulin, Z., & Ying, Z. (2011). A business process-driven approach for generating software modules. *Software - Practice and Experience, 41*(10), 1049 - 1071.

110. Zapletal, M., van der Aalst, W., Russell, N., Liegl, P., & Werthner, H. (2009). An Analysis of Windows Workflow's Control-Flow Expressiveness. *Web Services 2009. ECOWS'09 Seventh IEEE European Conference* (str. 200-209). Eindhoven: IEEE.

PRILOGE

KAZALO PRILOG

Priloga 1: Slovar tujk

Tabela 1: Slovar tujk

Slovenski izraz	**SLO kratica**	**ANGL kratica**	**Angleški izraz**
Management poslovnih procesov	MPP	BPM	Business Process Management
Poslovni proces	-	-	Business Process
Reinženiring poslovnih procesov	-	BPR	Business Process Reengineering
Delovni tok	-	-	Workflow
Upravljanje delovnega toka	-	-	Workflow Management
Orodje za Management poslovnih procesov	-	BPMS	Business Process Management System
Sistem za management delovnih tokov	-	WFMS	Workflow Management System
Izvajalnik delovnega toka	-	-	Workflow Runtime
Procesno usmerjeni informacijski sistemi	-	PAIS	Process-Aware Information Systems
Naravnanost k podatkom	-	-	Data-centric
Programske platforme	-	-	Software Platforms
Javanski stroj	-	-	Java Machine
Spletni servisi	-	-	Web Services
Od programiranja k montaži	-	-	From programming to assembling
Računalniško podprto načrtovanje	-	CAD	Computer Aided Development/Design
Modelsko vodene arhitekture	-	MDA	Model-Driven Development
Predstavitvena plast	-	-	Presentation Tier
Poslovna plast	-	-	Business Tier
Podatkovna plast	-	-	Data Tier
Večplastna arhitektura	-	-	Multi-Tier Architecture
Plast s poslovno logiko	-	-	Business Logic Layer
Plast za dostop do podatkov	-	-	Data Access Layer
Plast za splošne programske storitve	-	-	Common Application Services Layer
Storitveno usmerjena arhitektura	SOA	SOA	Service-Oriented Architecture
Storitveno vodilo	-	ESB	Enterprise Service Bus
Programski vmesnik	-	API	Application Programming Interface
Diagram primerov uporabe	-	-	Use Case Diagram
Kompozitni strukturni diagram	-	-	Composite Structure Diagram
Diagram razreda	-	-	Class Diagram
Zaporedni diagram	-	-	Sequence Diagram

se nadaljuje

nadaljevanje

Slovenski izraz	**SLO kratica**	**ANGL kratica**	**Angleški izraz**
Aktivnostni diagram	-	-	Activity Diagram
Objekt za opredelitev toka dogodkov	-	-	Flow Object
Dogodek	-	-	Event
Aktivnost	-	-	Activity
Podrejena aktivnost	-	-	Child Activity
Strukturirana aktivnost	-	-	Structured Activity
Prehod	-	-	Gateway
Povezovalni objekti	-	-	Connecting Objects
Tok sekvence	-	-	Sequence Flow
Tok sporočila	-	-	Message Flow
Asociacija	-	-	Association
Plavalne steze	-	-	Swim Lanes
Artefakt	-	-	Artifact
Podatkovni objekt	-	-	Data Object
Kontrolni vzorci	-	-	Control-flow Objects
Mreže delovnih tokov	-	-	Workflow Nets
Petrijeve mreže	-	-	Petri Nets
Označevalni jezik	-	-	Markup Language
Programiranje na visoki ravni	-	-	High-Level Programming
Programiranje na nizki ravni	-	-	Low-Level Programming
Skupni jezikovni izvajalnik	-	CLR	Common Language Runtime
Upravljanje izvajanja programske kode	-	-	Managed Code Execution
Upravljanje niti	-	-	Thred Management
Jezikovni prevajalnik	-	-	Language Compiler
Izvorna programska koda	-	-	Source Code
Ob pravem času	-	JIT	Just In Time
Izvajanje na osnovi dogodkov	-	-	Event-Driven Execution
Proces z dolgim časom izvajanja	-	-	Long-running Process
Mrežni servisi	-	-	Network Services
Orodje za razvoj delovnih tokov	-	-	Workflow Designer
Storitve za ohranjanje stanja	-	-	Persistence
Serializacija	-	-	Serialization
-	-	BPEL	Business Process Execution Language
-	-	WF	Windows Workflow Foundation
-	-	YAWL	Yet Another Workflow Language

se nadaljuje

nadaljevanje

Slovenski izraz	**SLO kratica**	**ANGL kratica**	**Angleški izraz**
-	-	BPMN	Business Process Modeling Notation
-	-	BPML	Business Process Modeling Language
Upravljanje stanja	-	-	State Management
Avtomat stanj	-	-	State-machine
Diagram toka podatkov	-	-	Flowchart
Razhroščevanje	-	-	Debugging
Spremenljivka	-	-	Variable
Parameter	-	-	Argument
Izraz	-	-	Expression
Porazdeljeno računalništvo	-	-	Distributed Computing
Sprožilec	-	-	Launcher
Podedovani informacijski sistemi	-	-	Legacy information systems
Vzorci delovnih tokov	-	-	Workflow Patterns
Vzorčni jezik	-	-	Pattern Language
Prestrezanje izjem	-	-	Exception Handling
Žeton	-	-	Token

Priloga 2: Vzorci delovnih tokov

Tabela 2: Vzorci delovnih tokov

VZOREC			OPIS
ŠT. VZORCA	**SLOVENSKI IZRAZ**	**ANGLEŠKI IZRAZ**	
WCP-1	Zaporedje	Sequence	Začetek izvajanja sledeče aktivnosti po izvršitvi predhodne aktivnosti v istem poslovnem procesu.
WCP-2	Vzporedni razcep	Parallel Split	Razcep poslovnega procesa v dve ali več vzporednih vej, ki se izvajajo istočasno (AND-split).
WCP-3	Sinhronizacija	Synchronization	Združitev (spojišče) dveh ali več vzporednih vej izvajanja procesa v eno po tem, ko se vse aktivnosti posameznih vej vzporednega izvajanja izvršijo (AND-join).
WCP-4	Ekskluzivni ALI	Exclusive Choice (XOR - split)	Na podlagi logične operacije (odločitve) se izvajanje po razcepu nadaljuje na eni izmed možnih vej poslovnega procesa (XOR-split).
WCP-5	Preprosta spojitev	Simple Merge (XOR - join)	Združitev (spojišče) dveh ali več vej poslovnega procesa v eno. Izvajanje se nadaljuje na enotni veji vsakič, ko se posamezna vhodna veja izvrši (XOR-join).
WCP-6	Večstranska izbira	Multiple Choice	Razcep veje procesa v dve ali več vej poslovnega procesa. Izvajanje se nadaljuje na eni ali več vejah, odvisno od rezultata logične odločitve v razcepu (OR-split).
WCP-7	Strukturirana sinhronizacija	Structured Synchronizing Merge	Združitev (spojišče) dveh ali več vej izvajanja, ki so se predhodno začele na istem razcepu. Združitev v eno se zgodi po tem, ko se vse aktivne veje izvajanja izvršijo.
WCP-8	Večstranka spojitev	Multi-Merge	Spojitev dveh ali več vej izvajanja v eno, izvajanje vsake veje se neodvisno nadaljuje v naslednji aktivnosti po spojitvi.

se nadaljuje

nadaljevanje

VZOREC			OPIS
ŠT. VZORCA	**SLOVENSKI IZRAZ**	**ANGLEŠKI IZRAZ**	
WCP-9	Strukturirani diskriminator	Structured Discriminator	Spojitev dveh ali več vej izvajanja v eno po tem, ko so se na nekem predhodnem razcepu ločile. Nadaljevanje procesa se zgodi takrat, ko se prva izmed vej izvajanja izvrši, doseže spojišče. Rezultati izvajanja preostalih vej procesa nimajo več vpliva na nadaljno izvajanje procesa. Diskriminacijski cikel se ponastavi takrat, ko se vse vhodne veje izvedejo.
WCP-10	Arbitražni cikli	Arbitrary Cycles	Zmožnost predstavitve ciklov v procesnem modelu, ki ima več kot en začetek ali konec. Nestrukturirana zanka (Unstructured loop).
WCP-11	Implicitna zaustavitev	Implicit Termination	Proces ali podproces se mora zaustaviti, ko ni več zaloge dela, ki bi ga bilo mogoče izvršiti v določenem trenutku ali kadar koli v prihodnosti.
WCP-12	Več instanc brez sinhronizacije	Multiple Instances without Synchronization	Znotraj posamezne instance procesa je lahko ustvarjenih več instanc posamezne aktivnosti. Instance so med seboj neodvisne ter se izvajajo istočasno, ko se posamezna instanca zaključi, sinhronizacija ni potrebna.
WCP-13	Število instanc, znano ob načrtovanju	Multiple Instances with a priori Design-Time Knowledge	Znotraj instance enega procesa se ob razvoju izvede določeno število instanc določene aktivnosti, ki se izvajajo istočasno in so med seboj neodvisne. Po zaključku instanc se te med seboj sinhronizirajo, šele nato je mogoče nadaljevanje procesa.
WCP-14	Število instanc, znano ob zagonu	Multiple Instances with a priori Run-Time Knowledge	Znotraj instance procesa se ob zagonu procesa izvede določeno število instanc določene aktivnosti, ki se izvajajo istočasno in so med seboj neodvisne. Po zaključku instanc se te med seboj sinhronizirajo, šele nato je mogoče nadaljevanje procesa.

se nadaljuje

nadaljevanje

VZOREC			OPIS
ŠT. VZORCA	**SLOVENSKI IZRAZ**	**ANGLEŠKI IZRAZ**	
WCP-15	Število instanc, neznano ob zagonu	Multiple Instances without a priori Run-Time Knowledge	Znotraj instance procesa se ob zagonu izvede neznano število instanc določene aktivnosti, ki je odvisno od več faktorjev, število instanc aktivnosti se lahko spreminja, dokler se že obstoječe instance aktivnosti izvajajo. Po zaključku vseh je potrebna sinhronizacija, šele nato je mogoče nadaljevanje procesa.
WCP-16	Odložena izbira	Deferred Choice	Točka v procesu, ko je v razcepu potrebna izbira samo ene od dveh ali več vej izvajanja, izbira pa je odvisna od prihodnjih dogodkov izvajanja v posameznih vejah. Ko se prva izmed vej izvajanja zaključi, je izvajanje vzporednih vej preklicano.
WCP-17	Prepleteno vzporedno usmerjanje	Interleaved Parallel Routing	Nekatere aktivnosti v procesu se izvajajo po vnaprej neznanem vrstnem redu, ki se določi v času izvajanja, hkrati pa je potrebno spoštovati logiko procesa. V določenem trenutku se lahko izvaja le ena aktivnost, vsaka aktivnost je lahko izvedena le enkrat.
WCP-18	Mejnik	Milestone	Izvajanje določene aktivnosti ali skupine aktivnosti je omogočeno le, ko stanje procesa to omogoča/dovoljuje. V primeru, da je instanca izvajanja procesa napredovala preko mejnika aktivacije izvajanj izbrane aktivnosti, ni več mogoča.
WCP-19	Preklic aktivnosti	Cancel Activity	Izvajanje aktivnosti je izvzeto iz delovnega toka, če pa se že izvaja, se to ustavi in, če je mogoče, izniči instanco aktivnosti.
WCP-20	Preklic dogodka	Cancel Case	Izvajanje celotne instance procesa in odvisnih podprocesov se zaustavi in prekliče ter zabeleži kot neuspešno zaključen proces.

se nadaljuje

nadaljevanje

VZOREC			OPIS
ŠT. VZORCA	**SLOVENSKI IZRAZ**	**ANGLEŠKI IZRAZ**	
WCP-21	Strukturirana zanka	Structured Loop	Zmožnost ponovljenega izvajanja aktivnosti ali podprocesa v nedogled. Pred začetkom ali ob koncu vsake zanke je prisotno preverjanje, ali je še prisoten pogoj za ponovno izvajanje.
WCP-22	Rekurzija	Recursion	Zmožnost aktivnosti, da izvede dodatno instanco same sebe ali določi izvajanje starševske aktivnosti.
WCP-23	Instantni sprožilec	Transient Trigger	Sprožitev izvajanja posamezne aktivnosti iz zunanjega okolja ali oddaljenega dela procesa, sprožilec je minljiv, če se izvajanje ne sproži v izbranem trenutku.
WCP-24	Vztrajnostni sprožilec	Persistenc Trigger	Sprožitev izvajanja aktivnosti iz zunanjega okolja ali oddaljenega dela procesa, sprožilec vztraja dokler se aktivnost ne zažene.
WCP-25	Preklic skupine	Cancel Region	Zmožnost izključitve skupine aktivnosti iz izvajanja. V primeru, da se katera izmed aktivnosti v skupini že izvaja, je izvajanje preklicano.
WCP-26	Preklic števila instanc skupine	Cancel Multiple Instance Activity	V primeru, da se v sklopu procesa izvaja več instanc skupine aktivnosti, pride do preklica izvajanja vseh instanc. Tiste, ki se še niso izvedle, se prekličejo, tiste, ki so se že izvedle, pa obstanejo.
WCP-27	Zaključek števila instanc aktivnosti	Complete Multiple Instance Activity	V primeru, da se v sklopu procesa izvaja več instanc skupine aktivnosti, zmožnost preklica še ne začetih instanc aktivnosti, prisiliti dokončanje že začetih instanc ter končna sinhronizacija zaključenih instanc pred nadaljevanjem procesa.

se nadaljuje

nadaljevanje

VZOREC			OPIS
ŠT. VZORCA	**SLOVENSKI IZRAZ**	**ANGLEŠKI IZRAZ**	
WCP-28	Blokirni diskriminator	Blocking Discriminator	V posamezni instanci procesa se pri spojitvi dveh ali več vej (ki so se predhodno v procesu razcepile) v eno, najprej izvrši ena veja, šele nato se omogoči nadaljevanje aktivnosti mimo spojitve naslednji veji procesa itd. Naslednja instanca procesa se omogoči takrat, ko se izvedejo vse še aktivne veje pred diskriminatorno spojitvijo.
WCP-29	Preklicni diskriminator	Cancelling Discriminator	V posamezni instanci procesa se pri spojitvi dveh ali več vej (ki so se predhodno v procesu razcepile) v eno, se ob izvršitvi prve veje izvajanje preostalih prekliče ter se omogoči izvajanje naslednje instance procesa.
WCP-30	Strukturirana delna spojitev	Structured Partial Join	V posamezni instanci procesa pri spojitvi M-števila vej (ki so se predhodno v procesu razcepile) v eno, se izvajanje mimo spojišča nadaljuje, ko se izvede N-število vej izvajanja procesa. Rezultati izvajanja preostalih (M-N) vej ne vplivajo na končno stanje procesa.
WCP-31	Blokirna delna spojitev	Blocking Partial Join	V posamezni instanci procesa, pri spojitvi M-števila vej (ki so se predhodno v procesu razcepile) v eno, se izvajanje mimo spojišča nadaljuje, ko se izvede N-število vej izvajanja procesa. Izvajanje še ne aktivnih vej, se prekliče, že aktivne veje se izvedejo do konca, nato se sprosti izvajanje sledeče instance procesa.
WCP-32	Preklicna delna spojitev	Cancelling Partial Join	V posamezni instanci procesa, pri spojitvi M-števila vej (ki so se predhodno v procesu razcepile) v eno, se izvajanje mimo spojišča nadaljuje, ko se izvede N-število vej izvajanja procesa. Izvajanje M-N vej procesa se prekliče ter se omogoči izvajanje naslednje instance procesa.

se nadaljuje

nadaljevanje

VZOREC			OPIS
ŠT. VZORCA	**SLOVENSKI IZRAZ**	**ANGLEŠKI IZRAZ**	
WCP-33	Posplošena spojitev	Generalized AND-Join	V posamezni instanci procesa, pri spojitvi M-števila vej (ki so se predhodno v procesu razcepile) v eno, se izvajanje mimo spojišča nadaljuje, ko se izvedejo vse veje posamezne instance procesa.
WCP-34	Statična delna spojitev za več instanc	Static Partial Join for Multiple Instances	Ob zagonu posamezne instance procesa se določi M število instanc določene aktivnosti. Pred sprožitvijo naslednje aktivnoste se mora izvršiti N število instanc aktivnosti. M-N instance se izvršijo do konca brez vpliva na končno stanje procesa.
WCP-35	Preklicna delna spojitev za več instanc	Cancelling Partial Join for Multiple Instances	Ob zagonu posamezne instance procesa se določi M-število instanc določene aktivnosti. Pred sprožitvijo naslednje aktivnoste se mora izvršiti N-število instanc aktivnosti. Izvajanje M-N instanc se prekliče.
WCP-36	Dinamična delna spojitev za več instanc	Dynamic Partial Join for Multiple Instances	Znotraj instance procesa se izvede ob zagonu neznano število instanc določene aktivnosti, ki je odvisno od več faktorjev, število potrebnih instanc se lahko spreminja, vse dokler se vse instance ne zaključijo. Po izvršitvi vsake instance se zgodi preverba pogoja (N/M) za sprožitev naslednje aktivnosti. Po sprožitvi sledeče aktivnosti se M-N aktivnosti izvršijo do konca brez vpliva na končno stanje procesa, dodatne instance aktivnosti se ne ustvarjajo.
WCP-37	Aciklična sinhronizacijska spojitev	Acyclic Synchronizing Merge	V posamezni instanci procesa, pri spojitvi dveh ali več vej (ki so se predhodno v procesu razcepile) v eno, se izvajanje mimo spojišča nadaljuje, ko se izvršijo vse aktivne veje procesa. Sinhronizacija pa se izvede za določeno število vej, ki je določeno ob razcepu.

se nadaljuje

nadaljevanje

VZOREC			OPIS
ŠT. VZORCA	**SLOVENSKI IZRAZ**	**ANGLEŠKI IZRAZ**	
WCP-38	Splošna sinhronizacijska spojitev	General Synchronizing Merge	V posamezni instanci procesa, pri spojitvi dveh ali več vej (ki so se predhodno v procesu razcepile) v eno, se izvajanje mimo spojišča nadaljuje, ko se izvršijo vse aktivne veje procesa.
WCP-39	Kritična izbira	Critical Selection	Nekatere aktivnosti dveh vzporednih vej izvajanja se ne morejo izvajati istočasno, temveč morajo počakati, da se ena ali druga »kritična izbira« zaključi.
WCP-40	Prepleteno usmerjanje	Interleaved Routing	Vsaka aktivnost iz skupine aktivnosti se mora izvršiti v poljubnem vrstnem redu vendar ne istočasno z ostalimi aktivnostimi. Šele ko se vse aktivnosti skupine izvršijo, se sproži nadaljevanje procesa.
WCP-41	Nitna spojitev	Thread Merge	V posamezni veji izvajanja, znotraj ene instance procesa, se lahko zgodi več niti skupine aktivnosti. Ko je doseženo M-število niti, se te združijo v eno in sprožijo nadaljevanje instance procesa.
WCP-42	Razdelitev niti	Thread Split	V posamezni veji izvajanja, znotraj ene instance procesa, se v določeni točki ustvari več niti izvajanja iste instance.
WCP-43	Eksplicitna zaustavitev	Explicit Termination	Proces ali podproces se mora zaključiti, ko doseže določeno stanje. Ko je stanje doseženo, je izvrševanje preostale zaloge dela preklicano, stanje procesa pa zabeleženo kot uspešno zaključeno.

Vir: N. Russel et al., Workflow Control-Flow Patterns: A Revised View, 2006, str. 1–80.